The differentiation strategy for proprietary software firms when Open Source software appears

Mingqing Xing[1,2]*

[1]*School of Economics and Management, Weifang University, Weifang, 261061, China*

[2]*Neural Decision Science Laboratory, Weifang University, Weifang, 261061, China*

Received 1 October 2014, www.cmnt.lv

Abstract

By extending the Hotelling model, this paper studies the software location and differentiation strategy for proprietary software firm when open source software emerges. It assumes that proprietary software firm pursues profit maximization and open source software can be freely available and mainly finds that: (i) Higher (resp. lower) the learning (maintenance or development) costs of open source software, smaller (resp. greeter) the software differentiation. (ii) the compatibility degree between open source and proprietary software affects the software differentiation strategy for proprietary software firm. (iii) the impact of network externalities or user's software development skills on proprietary software firm's software location and differentiation strategy may depend on the compatibility degree between open source and proprietary software.

Keywords: software differentiation, proprietary software, Open Source software, compatibility, network externality, hotelling model

1 Introduction

Since 1990s, the rapid development of open source software is a significant phenomenon in software industries, which shakes software industries dominated by proprietary software. For examples, Linux holds about 30% market and Microsoft's windows shares approximately 50% market share in server operating system market (Netcraft's survey, 2001) [1]. Apache (an open source software) shares more than 60% of web sits on the Internet in web server market, while Microsoft's Internet Information Service (a proprietary software) supports less than 30% market share (Netcraft's survey, 2006) [2]. Sendmail, an open source software, commands about 80% market share in the e-mail traffic market (Weber, 2004) [3]. Increasingly influential open source software changes the competition structure of software market and competitive strategies for software producer.

According to O'Reilly's definition [4], open source software is software, whose sources codes are allowed software developers to share, identify and correct errors, and redistribute. The rising and spreading of open source software arouse the attention of economic and management scholars. Some of them study the competition strategy in software industries, in which open source software and proprietary software coexist. Dalle and Jullien (2002) [5] analyze technological competition between open source and proprietary software by an interaction theory model. Raghunathan et al. (2005) [6] compare the optimal quality under proprietary software monopoly and duopoly competition between proprietary and open source software. Meng and Lee (2005) [7] consider the compatibility strategy between proprietary and open source software. Economides and Katsamakas (2006) [8] assume the software platform may be either proprietary or open source, but the applications are proprietary, and then examine platform competition in which each software platform supports multiple applications. Lin (2007) [9] investigates how user skills affect the market where proprietary software competes with open source software. Xing (2010) [10] studies quantity competition between open source and proprietary software. Cheng, Liu and Tang (2011) [11] examine the impact of network externalities on the competition between open source and proprietary software. Gramstad (2014) [12] develops a model to analyze competition between a commercial software producer and a free-of-charge open source software substitute in the presence of software piracy. From the perspective of technology, quality, compatibility, platform, user skill, quantity, network externality and piracy, the above literature analyzes competition between proprietary and open source software. However all of them do not consider the software location and differentiation strategy in a software industry when open source software appears.

As we all know, the software industry usually presents network externalities. The network externality in an industry is that the benefit that consumers obtain from purchasing one or several of its products depends on the number of other consumers that use the same or compatible products (Katz and Shapiro, 1985) [13]. This paper supposes that there coexist both open source software and proprietary software in a software market with network externalities. Through modifying the

* ***Corresponding author's*** e-mail: mqxing1979@163.com

Hotelling model, the paper studies the software location and differentiation strategy for proprietary software provider when open source software emerges and how the learning (maintenance or development) costs of open source software, user's software development skills, network externality and software compatibility influence the software location and differentiation.

The rest of this paper is organized as follows. Section 2 presents the basic model. Section 3 solves the model. Section 4 analyzes the model solution. Final part concludes this paper.

2 The basic model

There are two firms in a software market. One provides proprietary software (PS) and the other one provides open source software (OSS) (note that open source software is usually provided by an open source community). They are noted by firm 'p' and firm 'o' respectively. Borrowing ideas from Hotelling (1929) [14], the market is denoted by a unit linear interval '[0, 1]' and consumers are indexed by their preference for the software, which is measured by parameter 'x', and uniformly distributed with density 1 over the software market. Assume that proprietary software locates at 'x_p' and open source software locates at '1' in the market. The indirect utility for the generic user at x when he/she purchases proprietary and open source software are respectively given by

$$u_p = v + \alpha(q_p + k_p q_o) - p_p - \tau(x - x_p)^2, \qquad (1)$$

$$u_o = v + \alpha(q_o + k_o q_p) + \sigma q_o - l - \tau(1 - x)^2. \qquad (2)$$

In (1) and (2), v denotes the intrinsic utility (or quality) generated by proprietary software or open source software (for simplicity, this paper assumes this parameter is equal for proprietary and open source software). $\alpha(q_p + k_p q_o)$ and $\alpha(q_o + k_o q_p)$ represent the utilities from network externalities when using proprietary and open source software respectively, in which α is the intensity of network externalities. q_i ($i = p, o$) is firm i's software output (or network scale). k_p is the compatibility degree of proprietary software to open source software and k_o is the compatibility degree of open source software to proprietary software. p_p is the price of proprietary software (since open source software is usually free, $p_o = 0$); $\tau(x - x_p)^2$ and $\tau(1 - x)^2$ measure the user's utility losses associated with using proprietary and open source software that differs from his/her most preferred software respectively, in which τ denotes the degree of differentiation (we set $\tau = 1$ for simplicity). l is the learning (maintenance or development) costs if users buy open source software. σ is the degree of contribution for each user to the quality of open source software (or call it user's software development skills parameter).

To make sure the software market is fully covered, v is assumed to be big enough. Consumers have unit demand. That is, each consumer purchases unit software product from either firm 'p' or firm 'o'. Let $\bar{x}$ denote the preference of consumer who is indifferent between purchasing proprietary and open source software. $\bar{x}$ satisfies

$$\begin{aligned} &v + \alpha(q_p + k_p q_o) - p_p - (\bar{x} - x_p)^2 = \\ &v + \alpha(q_o + k_o q_p) + \sigma q_o - l - (1 - \bar{x})^2 \end{aligned} \qquad (3)$$

Since the software market is fully covered and consumers uniformly distribute, there are $q_p + q_o = 1$, $q_p = \int_0^{\bar{x}} dx$ and $q_o = \int_{\bar{x}}^1 dx$. Combining with (3), we obtain the demand functions

$$q_p = \frac{1 - x_p^2 - \alpha(1 - k_p) - \sigma + l - p_p}{2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma}, \qquad (4)$$

$$q_o = 1 - \frac{1 - x_p^2 - \alpha(1 - k_p) - \sigma + l - p_p}{2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma} \qquad (5)$$

According to (4) and (5), the profit functions for proprietary and open source software firms are given respectively by

$$\pi_p = p_p q_p = \frac{[1 - x_p^2 - \alpha(1 - k_p) - \sigma + l - p_p] p_p}{2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma}, \qquad (6)$$

$$\pi_o = p_o q_o = 0. \qquad (7)$$

Note that: since the marginal cost for software product is generally very low, this study assumes that the marginal costs for both proprietary software and open source software are equal to zero.

The timing of software location and pricing is as follows. In the first stage, proprietary software firm determines its software locations. In the second stage, firms set their software prices.

3 The model solutions

The model is solved by backwards induction. Thus, the price stage is analyzed firstly and then the location stage is studied.

The Price Stage.

Since open source software is free for users, we only need to solve the optimal price for proprietary software. According to (6), the first-order condition is

$$\frac{\partial \pi_p}{\partial p_p} = \frac{1 - x_p^2 + l - \alpha(1 - k_p) - \sigma - 2p_p}{2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma} = 0. \qquad (8)$$

Solving (8), the optimal price for proprietary software is given by

$$p_p^* = \frac{1 - x_p^2 + l - \alpha(1 - k_p) - \sigma}{2}. \tag{9}$$

To make sure p_p^* the unique optimal solution, p_p^* must satisfy the second-order condition, which requires

$$\frac{\partial^2 \pi_p}{\partial p_p^2} = \frac{-2}{2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma} < 0. \tag{10}$$

The inequality (10) holds if

$$2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma > 0. \tag{11}$$

This paper supposes the parameters meet the inequality (11).

Substituting (9) into (6), we derive proprietary software firm's profit function on its location variable x_p

$$\pi_p = \frac{[1 - x_p^2 + l - \alpha(1 - k_p) - \sigma]^2}{4[2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma]}. \tag{12}$$

The Location Stage.

Since open source software's location is exogenous ($x_o = 1$), we only need to solve the optimal location for proprietary software. Taking the derivative of profit function (12) with respect to x_p, and then setting it equal to zero

$$\frac{\partial \pi_p}{\partial x_p} = \frac{\begin{pmatrix} 1 - x_p^2 + l - \\ \alpha(1 - k_p) - \sigma \end{pmatrix} \begin{pmatrix} [3x_p^2 - 2(2 - \alpha(2 \\ -k_p - k_o) - \sigma)x_p + 1 \\ +l - \alpha(1 - k_p) - \sigma] \end{pmatrix}}{2[2(1 - x_p) + \alpha(k_p + k_o - 2) - \sigma]^2} = 0 \tag{13}$$

Solving (13), we derive four solutions that satisfy the first-order condition. According to the profit non-negative ($\pi_p \geq 0$) and the second-order condition ($\partial^2 \pi_p / \partial x_p^2 < 0$), three solutions are excluded. The optimal location of proprietary software is given by

$$x_p^* = \frac{\begin{pmatrix} 2 - \alpha(2 - k_p - k_o) - \sigma - \\ \sqrt{\begin{matrix}(2 - \alpha(2 - k_p - k_o) - \sigma)^2 \\ -3(1 + l - \alpha(1 - k_p) - \sigma)\end{matrix}} \end{pmatrix}}{3}. \tag{14}$$

This study supposes that the parameters can ensure the inequality $0 \leq x_p^* < 1$ (There have some parameter space areas meeting this inequality through the numerical analysis).

4 Comparative static analysis

This section analyzes how the parameters of software learning (maintenance or development) costs, user's software development skills, network externalities and software compatibility influence the software location (measured by x_p^*) and differentiation (measured by $1 - x_p^*$) strategy for proprietary software firm.

Taking the derivative of x_p^* with respect to l, we obtain

$$\frac{\partial x_p^*}{\partial l} = \frac{1}{2\sqrt{\begin{matrix}(2 - \alpha(2 - k_p - k_o) - \sigma)^2 \\ -3(1 + l - \alpha(1 - k_p) - \sigma)\end{matrix}}} > 0. \tag{15}$$

Thus, $\partial(1 - x_p^*)/\partial l = -\partial x_p^* / \partial l < 0$ and the following proposition is obtained.

Proposition 1: (i) Higher (resp. lower) the learning (maintenance or development) costs of open source software, proprietary software locating closer to (resp. far away) the open source software's location; (ii) Higher (resp. lower) the learning (maintenance or development) costs of open source software, smaller (resp. greeter) the software differentiation between proprietary and open source software.

The above proposition shows that the learning (maintenance or development) costs of open source software can affect the software location and differentiation strategy for proprietary software firms. For example, in operating system market, most of users are non-computer professional. They will bear high learning (maintenance or development) costs when use the open source operating system (Linux) and thus prefer to use the proprietary operating system (Windows). There are some similarities between Windows and Linux operating system. While in web server market most of users are computer professional. They will bear low learning (maintenance or development) costs when use the open source software (Apache) and thus not prefer to use the proprietary software (Microsoft's IIS). There exist very large differences between Apache and Microsoft's IIS.

According to Meng and Lee (2005) [7], there are four compatibility strategies for proprietary software to open source software: incompatibility, two-way compatibility, inward compatibility and outward compatibility. Now we analyze how network externalities and user's development capability affect the software location and differentiation strategy for proprietary software firm in the above four compatibility cases respectively.

When proprietary software and open source software are incompatibility ($k_p = k_o = 0$), this is the case that two types of software are fully not compatible with each other. For example, Windows, a proprietary software product, is incompatible with Linux, an open source software product.

Proposition 2: When $k_p = k_o = 0$, there are:

(i) $\frac{\partial x_p^*}{\partial \alpha} > 0$ and $\frac{\partial(1 - x_p^*)}{\partial \alpha} < 0$, if $l > \frac{\sigma}{2} - \frac{3}{16}$; $\frac{\partial x_p^*}{\partial \alpha} < 0$

and

$$\frac{\partial(1-x_p^*)}{\partial\alpha} > 0 \text{, if } l < \frac{\sigma}{2} - \frac{3}{16};$$

(ii) $\frac{\partial x_p^*}{\partial\sigma} > 0$ and $\frac{\partial(1-x_p^*)}{\partial\sigma} < 0$, if $l > \frac{1}{4} - \alpha$; $\frac{\partial x_p^*}{\partial\sigma} < 0$

and

$$\frac{\partial(1-x_p^*)}{\partial\sigma} > 0 \text{, if } l < \frac{1}{4} - \alpha.$$

Proof.

$$x_p^* = \frac{1}{3}[2-2\alpha-\sigma-\sqrt{(2-2\alpha-\sigma)^2-3(1-\alpha-\sigma+l)}]1$$

when $k_p = k_o = 0$. Taking derivatives of x_p^* with respect to α and σ respectively, we have

$$\frac{\partial x_p^*}{\partial\alpha} = \frac{1}{3}[-2+\frac{4(2-2\alpha-\sigma)-3}{2\sqrt{(2-2\alpha-\sigma)^2-3(1-\alpha-\sigma+l)}}]$$

and

$$\frac{\partial x_p^*}{\partial\sigma} = \frac{1}{3}[-1+\frac{2(2-2\alpha-\sigma)-3}{2\sqrt{(2-2\alpha-\sigma)^2-3(1-\alpha-\sigma+l)}}].$$

Thus,

$$\frac{\partial x_p^*}{\partial\alpha} > 0 \text{ and } \frac{\partial(1-x_p^*)}{\partial\alpha} < 0 \text{, if } l > \frac{\sigma}{2} - \frac{3}{16},$$

$$\frac{\partial x_p^*}{\partial\alpha} < 0 \text{ and } \frac{\partial(1-x_p^*)}{\partial\alpha} > 0 \text{, if } l < \frac{\sigma}{2} - \frac{3}{16}, \frac{\partial x_p^*}{\partial\sigma} > 0$$

and

$$\frac{\partial(1-x_p^*)}{\partial\sigma} < 0 \text{, if } l > \frac{1}{4} - \alpha, \frac{\partial x_p^*}{\partial\sigma} < 0$$

and

$$\frac{\partial(1-x_p^*)}{\partial\sigma} > 0 \text{, if } l < \frac{1}{4} - \alpha.$$

Proposition 2 demonstrates that, in the case of incompatibility, when the learning (maintenance or development) costs of open source software are high enough, the intensity of network externalities (or user's development skills) higher, the location of proprietary software closer to open source software and software differentiation smaller, and the opposite situation may appear when the learning (maintenance or development) costs of open source software are low enough. However, when user's development skills are low enough $(\sigma < 3/8)$ the intensity of network externalities higher, the location of proprietary software closer to open source software and software differentiation smaller no matter the level of open source software's learning (maintenance or development) costs. When the intensity of network externalities is high enough ($\alpha > 1/4$), the user's development skills higher, the location of proprietary software closer to open source software and software differentiation smaller no matter the level of open source software's learning (maintenance or development) costs.

When proprietary software and open source software are two-way compatibility ($k_p = k_o = 1$), this is the case that two types of software are compatible with each other. For example, Internet Explorer, a proprietary software product, is two-way compatible Mozilla, an open source software product.

Proposition 3: When $k_p = k_o = 1$, there are:

(i) $\frac{\partial x_p^*}{\partial\alpha} = 0$ and $\frac{\partial(1-x_p^*)}{\partial\alpha} = 0$;

(ii) $\frac{\partial x_p^*}{\partial\sigma} > 0$ and $\frac{\partial(1-x_p^*)}{\partial\sigma} < 0$, if $l > \frac{1}{4}$;

$\frac{\partial x_p^*}{\partial\sigma} < 0$ and $\frac{\partial(1-x_p^*)}{\partial\sigma} > 0$, if $l < \frac{1}{4}$.

Proof.

$$x_p^* = \frac{1}{3}[2-\sigma-\sqrt{(2-\sigma)^2-3(1-\sigma+l)}]$$

when $k_p = k_o = 1$. Taking derivatives of x_p^* with respect to α and σ respectively, we have

$$\frac{\partial x_p^*}{\partial\alpha} = 0 \text{ and } \frac{\partial x_p^*}{\partial\sigma} = \frac{1}{3}[-1+\frac{2(2-\sigma)-3}{2\sqrt{(2-\sigma)^2-3(1-\sigma+l)}}]$$

Therefore,

$$\frac{\partial(1-x_p^*)}{\partial\alpha} = 0, \frac{\partial x_p^*}{\partial\sigma} > 0 \text{ and } \frac{\partial(1-x_p^*)}{\partial\sigma} < 0 \text{, if } l > \frac{1}{4},$$

$$\frac{\partial x_p^*}{\partial\sigma} < 0 \text{ and } \frac{\partial(1-x_p^*)}{\partial\sigma} > 0 \text{, if } l < \frac{1}{4}.$$

The above proposition indicates that, in the case of two-way compatibility: (i) the software location of proprietary software and software differentiation are not affected by the network externalities; (ii) when the learning (maintenance or development) costs of open source software are high enough, the user's development skills higher, the software location of proprietary software closer to the location of open source software and software differentiation smaller, and the opposite situation appears when the learning (maintenance or development) costs of open source software are low enough.

When proprietary software and open source software are inward compatibility ($k_p = 1$ and $k_o = 0$), this is the case that proprietary software is compatible open source software, but open source software is not compatible with proprietary software. For example, Microsoft IIS, a proprietary web server, is inward compatible with Apache, an open source web server.

Proposition 4: When $k_p = 1$ and $k_o = 0$, there are:

(i) $\frac{\partial x_p^*}{\partial \alpha} \geq 0$ and $\frac{\partial (1-x_p^*)}{\partial \alpha} \leq 0$;

(ii) when $l > \frac{1}{4} - \alpha$, $\frac{\partial x_p^*}{\partial \sigma} > 0$ and $\frac{\partial (1-x_p^*)}{\partial \sigma} < 0$;

when $l < \frac{1}{4} - \alpha$, $\frac{\partial x_p^*}{\partial \sigma} < 0$ and $\frac{\partial (1-x_p^*)}{\partial \sigma} > 0$.

Proof.

$$x_p^* = \frac{1}{3}[2-\alpha-\sigma-\sqrt{(2-\alpha-\sigma)^2-3(1-\sigma+l)}].$$

When $k_p = 1$ and $k_o = 0$. Taking derivatives of x_p^* with respect to α and σ respectively, we have

$$\frac{\partial x_p^*}{\partial \alpha} = \frac{1}{3}[-1+\frac{2(2-2\alpha-\sigma)}{2\sqrt{(2-\alpha-\sigma)^2-3(1-\sigma+l)}}]$$

and

$$\frac{\partial x_p^*}{\partial \sigma} = \frac{1}{3}[-1+\frac{2(2-\alpha-\sigma)-3}{2\sqrt{(2-\alpha-\sigma)^2-3(1-\sigma+l)}}].$$

Because $0 \leq x_p^* < 1$, $1-\sigma+l \geq 0$.

Therefore, $\frac{\partial x_p^*}{\partial \alpha} \geq 0$ and $\frac{\partial (1-x_p^*)}{\partial \alpha} \leq 0$.

Moreover, $\frac{\partial x_p^*}{\partial \sigma} > 0$ and, $\frac{\partial (1-x_p^*)}{\partial \sigma} < 0$, if $l > \frac{1}{4} - \alpha$,

$\frac{\partial x_p^*}{\partial \sigma} < 0$ and $\frac{\partial (1-x_p^*)}{\partial \sigma} > 0$, if $l < \frac{1}{4} - \alpha$.

Proposition 4 shows that, in the case of inward compatibility: (i) the intensity of network externalities higher, the software location of proprietary software not closer to open source software and software differentiation not bigger; (ii) when the learning (maintenance or development) costs of open source software are high enough, the user's development skills higher, the software location of proprietary software closer to open source software and software differentiation smaller, and the opposite situation may appear when the learning (maintenance or development) costs of open source software are low enough. However, when the intensity of network externalities is high enough ($\alpha > 1/4$), the user's development skills higher, the software location of proprietary software closer to open source software and software differentiation smaller no matter the level of open source software's learning (maintenance or development) costs.

When proprietary software and open source software are outward compatibility ($k_p = 0$ and $k_o = 1$), this is the case that proprietary software is incompatible open source software, but open source software is compatible with proprietary software. This case is seldom in reality.

Proposition 5: When $k_p = 0$ and $k_o = 1$, there are:

(i) $\frac{\partial x_p^*}{\partial \alpha} = \frac{\partial x_p^*}{\partial \sigma} > 0$ and $\frac{\partial (1-x_p^*)}{\partial \alpha} = \frac{\partial (1-x_p^*)}{\partial \sigma} < 0$, if $l > \frac{1}{4}$

(ii) $\frac{\partial x_p^*}{\partial \alpha} = \frac{\partial x_p^*}{\partial \sigma} < 0$ and $\frac{\partial (1-x_p^*)}{\partial \alpha} = \frac{\partial (1-x_p^*)}{\partial \sigma} > 0$, if $l < \frac{1}{4}$.

Proof.

$$x_p^* = \frac{1}{3}[2-\alpha-\sigma-\sqrt{(2-\alpha-\sigma)^2-3(1-\alpha-\sigma+l)}]$$

When $k_p = 0$ and $k_o = 1$. Taking derivatives of x_p^* with respect to α and σ respectively, we have

$$\frac{\partial x_p^*}{\partial \alpha} = \frac{\partial x_p^*}{\partial \sigma} = \frac{1}{3}[-1+\frac{2(2-\alpha-\sigma)-3}{2\sqrt{(2-\alpha-\sigma)^2-3(1-\alpha-\sigma+l)}}]$$

Thus,

$\frac{\partial x_p^*}{\partial \alpha} = \frac{\partial x_p^*}{\partial \sigma} > 0$ and $\frac{\partial (1-x_p^*)}{\partial \alpha} = \frac{\partial (1-x_p^*)}{\partial \sigma} < 0$, if $l > \frac{1}{4}$,

$\frac{\partial x_p^*}{\partial \alpha} = \frac{\partial x_p^*}{\partial \sigma} < 0$ and $\frac{\partial (1-x_p^*)}{\partial \alpha} = \frac{\partial (1-x_p^*)}{\partial \sigma} > 0$, if $l < \frac{1}{4}$.

Proposition 5 indicates that, in the case of outward compatibility, when the learning (maintenance or development) costs of open source software are high enough, the intensity of network externalities (or user's development skills) higher, the software location of proprietary software closer to open source software and software differentiation smaller, and the opposite situation appears when the learning (maintenance or development) costs of open source software are low enough.

Through the above propositions, we find that proprietary software firm's software location and software differentiation strategy may be different in different cases of compatibility. Moreover, how network externalities and user's development skills affect proprietary software firm's software location and software differentiation may also be different when proprietary software and open source software implement different compatible strategies.

5 Conclusions

By modifying the Hotelling model, this study investigates the software location and software differentiation strategy for proprietary software firms when open source software appears. This paper assumes that proprietary software firm pursues profit maximization and open source software is free for users. When the software market is fully covered, the following conclusions are found: (i) Higher (resp. lower) the (maintenance or development) learning costs of open source software, smaller (resp. greeter) the software differentiation between open source and proprietary software; (ii) the compatibility degree between proprietary and open source software impacts the software differentiation; (iii) how network externalities and user's software development skills influence the software location and software differentiation strategy for proprietary software firm depends on the compatibility degree between proprietary and open source software.

Acknowledgement

We acknowledge the financial support from Natural Science Foundation of Shandong Province (No. ZR2013GL005), Social Science Planning Research Project of Shandong Province (No. 12CJRJ17), Shandong province soft science research plan (No. 2014RKB01165) and Shandong Higher School of Humanities and Social Science Research Projects (No. J13WF11).

References

[1] Netcraft, September 2001 web server survey. Available from: http://survey.netcraft.com/Survey/index-200109.html
[2] Netcraft, November 2006 web server survey. Available from: http://news.netcraft.com/archives/web_server_survey.html
[3] Weber S 2004 The Success of Open Source *Harvard University Press*
[4] O'Reilly T 1999 Lessons from Open-source Software Development *Communications of the ACM* 42 33-7
[5] Dalle J, Jullien N 2002 Open-source vs. Proprietary Software *Working paper*
[6] Raghunathan S, Prasad A, Mishra B K, Chang H 2005 Open Source versus Closed Source: Software Quality in Monopoly and Competitive Markets *IEEE Trans. Systems, Man, Cybernetics: Part A: Systems and Humans* 35 903-18
[7] Meng Z, Lee S Y 205 Open Source vs. Proprietary Software: Competition and Compatibility *Paper provided by EconWPA in its series industrial organization with number 0508008*
[8] Economides N, Katsamakas E 2006 Two-sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry *Management Science* 52 1057-71
[9] Lin L H 2008 Impact of User Skills and Network Effects on the Competition between Open Source and Proprietary Software *Electronic Commerce Research and Applications* 7 68-81
[10] Xing M Q 2010 The Quantity Competition between Open Source and Proprietary Software *Proceedings of International Conference on Information Management, Innovation Management and Industrial Engineering* 184-7
[11] Cheng H, Liu Y P, Tang Q 2011 The Impact of Network Externalities on the Competition between Open Source and Proprietary Software *Journal of Management Information Systems* 27 201-30
[12] Gramstad A R 2014 Piracy in Commercial vs. Open-Source Software Competition Available from: http://www.sv.uio.no/econ/english/research/news-and-events/events/conferences/2014/papers/gramstad_norio15may.pdf
[13] Katz M, Shapiro C 1985 Network Externalities, Competition, and Compatibility *American Economic Review* 75 424-40
[14] Hotelling H 1929 Stability in Competition *Economic Journal* 39 41-57

Author

Mingqing Xing, born in November, 1979, Shandong province, P.R. China.

Current position, grades: associate professor China of Weifang university.
University studies: graduated from College of Science of China Agriculture University in 2009 in China.
Scientific interest: software competition, two-sided market theory.
Publications: more than 30 papers published in various journals.
Experience: teaching experience of 5 years, 8 scientific research projects.