Transaction Cost Economics in Software Ecosystems: some empirical evidence

Wilco van Duinkerken

April 20, 2009

Chapter 5

Research

5.1 Research Questions

After the presentation of the Transaction Cost Economics , software ecosystems and the differences between software and traditional economic goods for which the Transaction Cost Economics was originally stated, curiosity leads to following research question:

Can Transaction Cost Theory be applied to explain the governance structure of relationships between buyers and suppliers of software components and services in Software Ecosystems?

If the Transaction Cost Economics is applicable to software ecosystems this could stimulate scholars and business in software ecosystems to think of partnerships and governance in software ecosystems using the lessons learned from the application of Transaction Cost Economics in non-software industries. If the Transaction Cost Economics is not the right theory to explain governance modes in software ecosystems, this would lead to the questions: "What is a way to explain and or guide governance mode in Software Ecosystems?".

In order to answer the main research question, six sub-questions are defined:

1. Which governance structures can be seen in Software Ecosystems?
2. How to adapt the Transaction Cost Theory concepts to Software Ecosystems?
   (a) How do the governance structures in Software Ecosystems map to the Transaction Cost Economics governance structures?
   (b) How to translate the classic concept of *Asset Specificity* to Software Ecosystems?
   (c) How to translate the classic concept of *Recurrence* to Software Ecosystems?
   (d) How to translate the classic concept of *Uncertainty* to Software Ecosystems?
   (e) How to translate the classic concept of *Production Costs* to Software Ecosystems?

In order to clarify the research question the remainder of this chapter will provide definitions and clarifications of the separate parts of our research question. This includes the conceptual research framework and the presentation of the hypotheses tested in this thesis research.

5.2 Definitions

5.2.1 Transaction Cost Economics

The first major definition encountered in the main research questions is "Transaction Cost Economics ". Since an extensive overview of Transaction Cost Economics is given in the literature study of this thesis, it suffices to present a brief summary of the definitions of Transaction Cost Economics .

The independent variable of the Transaction Cost Economics is the Governance Structure which is determined by the dependent variables Asset Specificity, Uncertainty and Recurrence. These concepts are defined as:

Governance structure An institutional framework in which the integrity of a transaction of related set of transactions is decided [49, p. 11]. In the light of Transaction Cost Economics this thesis looks more specifically at the degree of hierarchical or market nature of the governance structure.

Asset specificity The degree to which an asset can be redeployed to alternative uses and by alternative users without sacrifice of productive value [49, p. 59].

Uncertainty Classic Transaction Cost Economics derives three categories of uncertainty: (i) External uncertainty being statistical risks or state-contingent (ii) Internal uncertainty which arises "from lack of communication, that is from one decision maker having no way of finding out the concurrent decisions and plans made by others"[27, p. 147] and (iii) behavioral uncertainty which containts "strategic nondisclosure, disguise, or distortion of information"[49] by transaction partners.

Recurrence The amount of times a transaction is repeated [50].

5.2.2 Software Ecosystem

The second major definition encountered is "Software Ecosystem". Throughout this thesis we will use the definition of Jansen, Finkelstein and Brinkkemper:

Software Ecosystem a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts.

5.2.3 Software Services and Components

The unit of analysis of Transaction Cost Economics is a single transaction. In the light of Software Ecosystems the transaction between two software vendors is centered around software components and software services. In literature and amongst scholars there seems to be no unified definition of the term software component. A rather old, but interesting discussion about this subject can be read in Broy et. al (1998).

Since literature didn't provide a solid definition of a software component suitable for this research, it was decided to come up with a tailored definition based on the definition of Messerschmitt and Szyperski's [31, p. 247-250]. They define a software component as: "A reusable module suitable for composition into multiple applications". A definition of module is not given. Messerschmitt and Szyperski continue their discussion about software components by providing a list of properties that in general should apply to software components: Multiple-use, Non-context-specific, composable, encapsulated and unit of independent deployment and versioning. The list of properties, their descriptions and the rationale behind the properties given by Messerschmitt and Szyperski is, although it seems a reasonable lists of properties at first, a complete utopia.

Table 5.1 presents the original properties, description and rationale of Messerscmitt and Szypersky. The last column of the table contains the reasons why the property, description and/ or rationale is not useful in this thesis.

In this thesis a less restricted definition of software components is used:

Software Component a bundle of software functions accessible through a single interface or carrying a single name which is or can be used as an element in other software packages but of which the core functionality is developed separate from these packages.

Software Service A software service is a software component accessible via communications outside the users native environment.

User can be both human and non-human

Native environment changes depending on the user. If the user is human, the native environment is most probably his or her own computer. If the user is a software package, the native environment consists of the compiled or interpreted code.

5.3 Research Framework

5.3.1 Production Costs

Let ΔPC denote the difference in the cost of producing the software component at the market, PC_m and producing it internally, PC_i.

$$\Delta PC = PC_m - PC_i \tag{5.1}$$

PC_m is determined by the license and service fees of the component. If the license or service fee is recurrent PC_m is based on a 3-year contract period. If the license fees are based on a fee-per-transaction basis PC_m is calculated based on 3-years of forecasted use. The period of 3 years is an arbitrary choice based on the idea that an internal piece of software will survive 3 years within a product software company. Furthermore the Dutch tax offices allow companies to devaluate software over a period of 3-5 years if this software exceeds an initial investment of xxx euros.

PC_m does not include any implementation costs. Implementation costs are seen as dedicated asset investments and are filed under asset specificity, increasing the transaction costs. Examples

Property	Description	Rationale	Response
Multiple Use	Able to be used in multiple projects.	Shared development cost over multiple uses.	If one person can implement a component, more people can. So once it is done once, the multiple use property is statisfied. This doesn't mean that a component which is tailor made for single usage should not be called component. Therefore sharing the development costs over multiple uses it not the right reasoning behind the multiple usage.
Non-context-specific	Designed independently of any specific project and system context.	By removing dependence on system context, more likely to be general and broadly usable.	A lot of components start their live as a integrated part of a bigger system and become more abstract and non-context-specific later on. If early on a company decides to use the (open source) component, tailor it to specific needs and start using it. It could still be considered a software component, although both in the beginning and the end the component is very context-specific.
Composable	Able to be composed with other components.	High development productivity achieved through assembly of components.	With middleware technologies and standards, almost every piece of software can be tied to another piece of software.
Encapsulated	Only interfaces are visible and the implementation cannot be modified.	Avoids multiple variations; all uses of a component benefit from a common maintenance and upgrade effort.	There are a lot of open source software components, which by open source nature, get forked, changed and tailored, spinning out into multiple variations. These forks should all be considered a component.
Unit of independent deployment and versioning	Can be deployed and installed as an independent atomic unit and later upgraded independently of the remainder of the system.	Allows customers to perform assembly and to mix and match components even during the operational phase, thus moving competition from the system to the component level.	Full component independence and hot swaps of pieces of software for the biggest part remain impossible, especially in compiled software.

Table 5.1: Properties that distinguish a component [31, p. 248]

Bibliography

[1] Ron Adner. Match your innovation strategy to your innovation ecosystem. *Harvard Business Review*, April 2006.

[2] Armen A. Alchian and Harold Demetz. Production, information costs, and economic organization. *American Economic Review*, LXII, December 1972.

[3] Soon Ang and Detmar W. Straub. Production and transaction economies and is outsourcing: A study of the u.s. banking industry. *MIS Quarterly*, 22(4):535–552, 1998.

[4] Ulli Arnold. New dimensions of outsourcing: a combination of transaction cost economics and the core competencies concept. *European Journal of Purchasing Supply Management*, 6:23–29, 2000.

[5] Benoit A. Aubert, Suzanne Rivard, and Michel Patry. A transaction cost approach to outsourcing behavior: Some empirical evidence. *Information Management*, (30):51–64, 1996.

[6] Benoit A. Aubert, Suzanne Rivard, and Michel Patry. A transaction cost model of it outsourcing. *Information Management*, (41):921–932, 2004.

[7] Boris Beizer. Software is different. *Annals of Software Engineering*, 10:293–310, 2000.

[8] J. Bosch. The rise of software ecosystems on the web. 2009.

[9] Sjaak Brinkkemper, Ivo van Soest, and Slinger Jansen. *Information Systems Development*, chapter Modeling of Product Software Businesses: Investigation into Industry Product and Channel Typologies, pages 1–19. Springer US, 2009.

[10] E. Brynjofsson. The productivity paradox of information technology. *Communications of the ACM*, 36(12):66–77, 1993.

[11] E. Brynjofsson and Lorin M. Hitt. Beyond the productivity paradox: Computers are the catalyst for bigger changes. *Communications of the ACM*, 41(8):49–55, August 1998.

[12] Ronald H. Coase. The nature of the firm. *Economica N.S.*, (4):386–405, 1937.

[13] M.A. Cusumano. *The Business of Sofware: What Every Manager, Programmer and Entrepreneur Must Know to Succeed in Good Times and Bad.* Free Press, New York, NY, 2004.

[14] Barbara Farbey and Anothny Finkelstein. Software acquisition: a business strategy analysis. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, pages 67–83. IEEE Computer Society, August 2001.

[15] Lucia Gao and Bala Iyer. Analyzing complementarities using software stacks for software industry acquisitions. *J. Manage. Inf. Syst.*, 23(2):119–147, 2006.

[16] Lucia S. Gao and Bala Iyer. Analysing complementarities using software stacks for software industry acquisitions. *Journal of Management Information Systems*, 32(2):119–147, 2006.

[17] Lucia S. Gao and Bala Iyer. Partnerships between software firms: Is there value form complementarities? In *Proceedings of the 41st Hawaii International Conference on System Sciences*, 2008.

[18] I. Heitlager, S. Jansen, R. Helms, and S. Brinkkemper. Understanding the dynamics of product software development using the concept of coevolution. *Software Evolvability, IEEE International Workshop on*, 0:16–22, 2006.

[19] Bala Iyer, Chi-Hyon Lee, and David Dreyfus. Competing in the era of emergent architecture: The case of packaged software industry. *Hawaii International Conference on System Sciences*, 0:209b, 2007.

[20] Bala Iyer, Chi-Hyon Lee, and N. Venkatraman. Managing in a 'small world ecosystem': Lessons from the software sector. *California Management Review*, 48(3):28–47, 2006.

[21] Bala Iyer, Chi-Hyon Lee, N. Venkatraman, and Dan Vesset. Monitoring platform emergence: Guidelines from software networks. *Communications of the Assoction for Information Systems*, 18, December 2006.

[22] S. Jansen, A. Finkelstein, and S. Brinkkemper. A sense of community: A research agenda for software ecosystems. In 31st International Conference on Software Engineering, New and Emerging Research Track, 2009.

[23] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. Component assembly mechanisms and relationship intimacy in a software supply network. 15th International Annual EurOMA Conference Special Interest Session on Software Supply Chains, 2008.

[24] Slinger Jansen, Sjaak Brinkkemper, Ivo Hunnik, and Cetin Demir. Pragmatic and opportunistic reuse in innovative start-up companies, 2008.

[25] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. Providing transparency in the business of software: A modelling technique for software supply networks. In *roceedings of the 8th IFIP Working Conference on Virtual Enterprises*, pages 677–686, 2007.

[26] Hans-Bernd Kittlaus and Peter N. Clough. *Software Product Management and Pricing: Key Success Factors for Software Organizations*. Springer-Verlag Berlin, Heidelberg, 2009.

[27] Tjalling Koopmans. *Three Essays on the State of Economic Science*. McGraw-Hill, New York, 1957.

[28] Jeffrey T. Macher and Barak D. Richman. Transaction cost economics: An assessment of empirical research in the social sciences. *Business and Politics*, 10(1):1–63, 2008.

[29] Scott E. Masten. The organization of production: Evidence from the aerospace industry. *Journal of Law and Economics*, 27:401–417, 1984.

[30] Scott E. Masten, Jr. James W. Meehan, and Edward A. Snyder. The cost of organization. *The Journal of Law, Economics Organization*, 7(1):1–25, 1991.

[31] David G. Messersmitt and Clemens Szyperski. *Software Ecosystem: understanding an indispensable technology and industry*. The MIT Press, Cambridge, Massachusetts, London, England, 2003.

[32] David C. Mowery and Richard R. Nelson, editors. *Sources of Industrial Leadership*. Cambridge University Press, USA, 1999.

[33] K. Nam, S. Rajagopalan, H.R. Rao, and A. Chaudhury. A two-level investigation of information systems outsourcing. *Communications of the ACM*, 39(7):37–44, 1996.

[34] Douwe Postmus and Gert Kruithof. Supply chain management in the erp industry. 2008.

[35] Douwe Postmus and Theo Dirk Meijer. Aligning the economic modeling of software reuse with reuse practices. *Information and Software Technology*, 50:753–762, 2008.

[36] Douwe Postmus, Theo Dirk Meijer, and Hans Wortmann. A typology of product delivery strategies in the software industry.

[37] Slinger Jansen Wilfried Rijsemus. Balancing total cost of ownership and cost of maintenace within a software supply network. In *Proceedings of the IEEE International Conference on Software Maintenance*, Philadephia, PA, USA, September 2006.

[38] Aric Rindfleisch and Jan B. Heide. Transaction cost analysis: Past, present, and future applications. *Journal of Marketing*, 61(4):30–54, October 1997.

[39] Michael H. Riordan and Oliver E. Williamson. Asset specificity and economic organization. *International Journal of Industrial Organization*, 3:365–378, 1985.

[40] Howard A. Shelanski and Peter G. Klein. Empirical research in transaction cost economics: A review and assessment. *Journal of Law and Economics*, 11(2):335–361, 1995.

[41] Lucia Silva and Bala Iyer. Using software stacks to explain complementaries: The case of mergers and acquisitions in the software industry. In *System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on Software Systems*, volume 8, 2006.

[42] Herbert Simon. *Administrative Behavior*. Macmillan, New York, 1961.

[43] G. Slater and D.A. Spencer. The uncertain foundations of transaction cost economics. *Journal of Economic Issues*, 34(1):61–87, 2000.

[44] Aidan Vining and Steven Globerman. A conceptual framework for understanding the outsourcing decision. *European Management Journal*, 17(6):645–654, 1999.

[45] Oliver E. Williamson. The vertical integration of production: Market failure considerations. *American Economic Review*, LXI(2):112–23, May 1971.

[46] Oliver E. Williamson. Transaction-cost economics: The governance of contractual relations. *The journal of law economics*, 22(2):233–262, transaction cost theory economics 1979.

[47] Oliver E. Williamson. The economics of organization: The transaction cost approach. *The American Journal of Sociology*, 87(3):548–577, 1981.

[48] Oliver E. Williamson. Comparative economic organization: The analysis of discrete structural alternatives. *Administrative Science Quarterly*, 36(2):269–296, 1991.

[49] Oliver E. Williamson. *The Mechanisms of Governance*. Oxford University Press, New York, 1996.

[50] Oliver E. Williamson. The teory of the firm as governance structure: From choice to contract. *Journal of Economic Perspectives*, 16(3):171–195, Summer 2002.

[51] Oliver E. Williamson and Scott E. Masten. *The Economics of Transaction Costs*. Edward Elgar Publishing ltd., Massachusetts, USA, 1999.

[52] Lai Xu and Sjaak Brinkkemper. Concepts of product software: Paving the road for urgently needed research. In *CAiSE Workshops*, volume 2, pages 523–528, 2005.