

Abstract

La mise à niveau d'un système est un point important de la prise de décision du point de vue d'une organisation utilisatrice d'un logiciel intégré. L'étape de son cycle de vie dans lequel se trouve un produit logiciel, que ça soit le développement initial, l'évolution, la maintenance, l'élimination progressive ou la fin de vie, influence la décision concernant un changement de version. L'analyse de la littérature suggère que le timing de la mise à niveau est influencé par la phase du cycle de vie du logiciel, l'existence d'une version appropriée, et l'écart entre le système en production et la nouvelle version. Le meilleur moment pour une mise à niveau est celui où le logiciel est dans sa phase d'évolution et que le vendeur développe activement de nouvelles versions. Des conflits surgissent à cause d'intérêts divergents du client et du vendeur. L'intérêt des vendeurs est de produire fréquemment de nouvelles versions et que les clients suivent le programme de mise à niveau afin de générer des revenus. L'intérêt d'un client est de mettre à niveau son système lorsque cela est nécessaire vers une version qui satisfait les besoins de l'organisation et requiert le minimum de changements dans celle-ci. Ces intérêts divergents, du point de vue du client, nécessitent de faire attention aux changements qui ont lieu dans le cycle de vie et le développement du produit, spécialement dans les étapes de maintenance et de fin de vie.

Mots-clefs: système ERP, produit logiciel, changement de version, mise à niveau, timing

Abstract

System upgrade is an important decision making point from packaged system user organisation's point of view. The life cycle stage of a software product – whether it is at the initial development, evolution, servicing, phase-out or closedown stage – influences the decision concerning version change. The literature analysis suggests that the timing version change is influenced by the life cycle phase of the software product, the existence of a suitable version, and the gap between the system in use and the newest version. The best time for system upgrade is when the software product is on its evolution phase and the vendor is actively developing new versions. Conflicts arise due differing interests of customer and vendor. The interest of software vendors is to produce new versions frequently and to have the customers to follow the upgrading program in order to create revenue. The interest of a customer is to upgrade their system when needed to a version that meets the organisation's needs, and requires minimum changes in the organisation. These differing interests, from customer point of view, necessitate alertness in paying attention to the changes taking place in product life cycle and product development, especially in servicing and phase-out stages.

Key-words: ERP system, software product, version change, up-grading, timing

Considérations sur le timing de la mise à niveau d'un système de gestion intégré (ERP) à la lumière des intérêts du vendeur et du client communication

Considerations on ERP system upgrade timing in the light of vendor's and customer's interests

Irja KANKAANPÄÄ

Postgraduate student

University of Jyväskylä

Department of computer science and information systems

P.O. Box 35, Agora

FIN-40014 Jyväskylä, Finland

+358 14 260 3870

irja.k.kankaanpaa@jyu.fi

Petri MAARANEN

Lecturer

University of Jyväskylä

Department of computer science and information systems

P.O. Box 35, Agora

FIN-40014 Jyväskylä, Finland

+358 14 260 3015

maaranen@cc.jyu.fi

Introduction

System upgrade is an important decision making point from both system vendor's and system user's point of view. System vendor has to produce new versions in order to stay in markets. System user, on the contrary, chooses which version best supports its business functions and when is the proper time for system upgrade. Nowadays, system vendors produce new versions on the markets faster than the customers can adopt them (Sawyer 2001). The conflicting interests of software vendor and customer are challenging for the user organization's IS management. They are particularly interesting in case of version upgrade.

The customer commits to a relationship with the system vendor for a relatively long period of time when purchasing an ERP system (Markus and Tanis 2000, Verville, Bernadas et al. 2005). Thus, shared interests and mutual trust are important factors influencing the quality of cooperation between vendor and customer. Upgrade decisions are interesting because, on one hand, they should be established on a trusting vendor-customer relationship yet on the other hand they put the stakeholders in a position of where interests conflict.

System upgrades are becoming an integral part of IS management yet it seems that timing of a version change is defined in unsystematic manner in user organisations (Mukherji, Rajagopalan et al. 2006). Investments in new systems have been receiving plentiful research interests in the past while the research on system upgrade has been overlooked (Mukherji, Rajagopalan et al. 2006). The timing of the packaged system upgrade has been a neglected research area particularly in ERP research (Ng 2001).

This paper draws together a theoretical contribution from the existing literature on both consumer-vendor relationship and upgrade decisions including upgrade timing of packaged systems. The goal is to identify the critical touch-points of packaged systems life cycle from consumer's and from vendor's perspective, i.e. the critical preconditions that enable successful version upgrade. The timing of system upgrade and the related conflicting interests between vendor and customer are studied. As a result a set of guidelines is derived for the benefit of the consumer planning system upgrade.

The rest of the paper is organised as follows. The life cycle of software product is described in section 1. The vendor-customer relationship is studied in section 2. An analysis of ERP system evolution, including the timing and version selection is given in section 3. Discussion is provided in section 4 and summary is given in section 5.

1. Software product's life cycle

Commercial software products, particularly socio-technical applications, are specialized and increasingly diverse (Messerschmitt and Szyperski 2003). Typical off-

the-shelf products are Enterprise Systems (ES) including Enterprise Resource Planning (ERP) systems and Customer Relationship Management (CRM) systems (Wagner and Newell 2007), and collaborative applications such as Lotus Notes (Karsten 1999).

The software product's development process can be divided into two distinct perspectives, namely consumer's perspective and producer's perspective (Sawyer 2001). Vendor's interest is to develop software products for the markets, while the customer assembles a coherent IS out of software pieces produced by a system vendor (Sawyer 2001). The distinction between these two development perspectives is important because they are each other's complements and together form the complete IS life cycle. Individually, though, they present only "part of the truth". Thus, their difference should be recognised and analysed.

1.1. Consumer's perspective

The user organization selects a product from the markets that best corresponds to its needs and then uses consultants to install and adopt it in order to turn it into an IS (Sawyer 2001). The customer performs planning, selection, initiation, system-needs analysis, product-feature analysis, gap-fit comparison, installation and support (Sawyer 2001). These steps are particularly focused on buying a system, not developing one (Sawyer 2001). The customers do not define requirements based on their own needs but rather develop a plan on how to link the "perceived organisational needs to known and emerging software products" (Sawyer 2001).

Esteves and Pastor (1999) present a wider view on ERP system life cycle comprising of the following phases: adoption decision phase, acquisition phase, implementation phase, use and maintenance phase, evolution phase, and retirement phase. This process includes the activities after traditional use and maintenance, i.e. evolution and retirement. Cooper and Zmud (1990) suggest that packaged software implementation follows the process of initiation, adoption, adaptation, acceptance, routinisation, and infusion. Their model emphasises the implementation of a new system in organisational context. Markus and Tanis (2000) proposed a four stage model consisting of chartering, project, shakeout and onward and upward phases (Markus and Tanis 2000). The model of Esteves and Pastor (1999) is the most inclusive with respect to system life cycle and thus it is selected as the basis in this study (from consumer perspective). It is complemented with models of Cooper and Zmud (1990) and Sawyer (2001) when necessary. The combined consumer oriented product life cycle consists of the following phases: initiation, adoption, implementation, use and maintenance, evolution phase, and retirement. The life cycle phases of a software product and related characteristics from customer's viewpoint are summarised in TABLE 1.

TABLE 1. IS life cycle from customer's view

Phase	Characteristic	Meaning
Initiation	Need-pull or technology-push force	Trigger for acquisition
Adoption	Investment decision	"Marriage with the vendor"
Implementation	Successful technical and social implementation, adaptation to and acceptance of the system	Precondition for use and system success
Use and maintenance	Routine use, routine maintenance	Steady operation
	More capabilities needed or insufficient functionality	Trigger for system upgrade
Evolution	New capabilities integrated, version change	Starting point for the implementation of the new version
Retirement	End of vendor support, problems with maintenance	Trigger for abandoning the system

1.2. Vendor's perspective

According to Rajlich and Bennett (2000) five distinct stages can be identified in the development process of a software product. Initial development refers to a phase where the system's first functioning version is built from scratch. Software team expertise and coherent system architecture are the foundation stones for long life and successful further evolution of the system.

A packaged system seldom remains unchanged over time. Instead, it tends to evolve. During evolution "the capabilities and functionality of the system" are extended in order to meet the needs of the users. The system undergoes a sequence of iterative changes triggered by customer demands, competitive pressure or legislative actions. (Rajlich and Bennett 2000) Evolution of packaged software is manifested through versions. "At certain intervals a company completes a version of its software and releases it to customers" (Rajlich and Bennett 2000). A "version refers to strategic changes during evolution" (Rajlich and Bennett 2000). A release "refers to servicing patches" (Rajlich and Bennett 2000). System upgrade refers to the action of a consumer to acquire a new version of the packaged system it uses.

New versions are released at regular intervals until the software reaches the servicing phase. Servicing means that evolution is no longer possible or it is increasingly difficult and expensive. Vendor no longer redevelops the product but provides corrective maintenance, i.e. service patches. At this phase the coherent architecture easily becomes an obstacle for non-evolutionary changes. Corrective patches degrade the architecture and consequently speed up software decay (Rajlich and Bennett 2000).

Phase-out takes place when the software has decayed to a state when it is decided that no more servicing is going to

be done. Vendor seeks to generate revenue from the system as long as possible without changing it. Since the servicing is stopped, the users, when facing difficulties have to work their way around them (Rajlich and Bennett 2000). In the end, is the closedown. The vendor finally withdraws the system from the markets and the users are induced to a replace it with another system (Rajlich and Bennett 2000). The life cycle stages of a software product and its characteristics from vendor's viewpoint are summarized in TABLE 2.

TABLE 2. Software life cycle from vendor's view, based on Rajlich and Bennett (2000)

Phase	Characteristic	Meaning
Initial development	Software team expertise, system architecture	Foundation for future evolution
Evolution	Software team expertise, system architecture	Precondition for evolution
	Customer demands, competitive pressure, legislation, changes in business practices or in operation environment	Trigger for evolution
	Market demand, buoyant sales, good revenue	Driver for evolution
Servicing	System architecture degradation	Cause of software decay
Phase-out	Software decay	Reason for ending the support
Closedown	Contracts	Liability towards outsourced software
	Software decay, legacy system	Reason for closedown

As we have seen, it would be of utmost importance for a customer to understand the life-cycle phase of a vendor's product development. This potential risk factor with vendor-customer relationship will be discussed in the next section.

2. Vendor-customer relationship

Vendor-customer relationship becomes critical as the size and criticality of the target system increases. "Organizations that purchase an enterprise system enter into long-term relationships with software vendors, even for five to ten years (Verville et al., 2005). Hence, the relationship between customer and vendor should be strategic in a way that the vendor enhances the user organisation's competitiveness and efficiency through the ERP system (Somers and Nelson, 2004). A good partnership will promote successful system implementation (Somers and Nelson, 2004). Trust and open climate for co-operation are prerequisites for conflict-free relationship (Verville et al., 2005). In this partnership, the consumer is, however, dependent on the vendor (Markus and Tanis, 2000).

2.1. Vendor's interests

Throughout the software life cycle, the vendor's main interest is to make profit. Initial development is expensive with potentially no return. Therefore it is in the vendor's interest to ship the product as fast as possible in order to "generate revenue and to beat any competition" (Rajlich and Bennett, 2000). Particularly in the beginning of a software product life cycle the monopolist software producers use the tactics of "leveraging incompatibility" between the existing versions that are in use and new versions in order to pressure the customers to update their software (Mehra and Seidmann, 2006).

Once a product gains a steady market position it is redeveloped (evolved) to meet customer needs. New versions are produced at frequent intervals and customers are induced to purchase them. In order to ensure sales SAP, for instance, « recommends that the project team follow the upgrade of SAP releases and should consider the adoption of new ones » (Esteves and Pastor, 2005). Wei et al. (2007) have published some interesting insights about vendors pricing policies related to the attitudes towards the software application of customer population.

As soon as the sales drop it is economical to shift to servicing (Rajlich and Bennett, 2000). Servicing is something that the vendor may consider worth of outsourcing servicing because it does not require the company expertise (Rajlich and Bennett, 2000). At servicing stage the vendor avoids major changes in the software because at that stage they are expensive. Instead of coherent evolutionary changes they prefer to provide wrapping or patches for customers (Rajlich and Bennett, 2000). At phase-out stage, it is in the vendor's interests to try to get revenue from the system with no efforts (changes) as long as possible (Rajlich and Bennett, 2000) before withdrawing the product from the markets.

2.2. Customer's interests

Markets largely guide the IS investment decisions and company's procurement strategy (Narasimhan, Talluri et al. 2006). When getting a system upgrade a company is interested in getting a new version that corresponds to their business needs. The customer is interested in the quality of the product, how the product is delivered, how responsive the vendor is, and how innovative the vendor is. All this sums up in how effective the vendor is in meeting the changing needs of customers. (Narasimhan, Talluri et al. 2006) Typically, the customer wants to be able to influence on the changes of the software and expects that the vendor produces new versions that meet their needs better than the previous version. A company facing a sudden change in the business environment that requires changes in the information systems is relying on the vendor to be able to react on the change request and modify the software accordingly.

It is in the interests of a company to upgrade only when upgrading is needed. Frequent or unnecessary upgrades,

even if they were technologically feasible, cause problems (Esteves and Pastor 2005) by consuming staff resources and IT budget in vain (Mukherji, Rajagopalan et al. 2006). When adopting a new version, the customer wants to get the version change done as efficiently as possible. It is important to recover from the upgrade and return back to the normal use level as quickly as possible in order to avoid drop in productivity (Mukherji, Rajagopalan et al. 2006).

Typically, a customer acquires "products that are in different stages of their respective product life cycles" (Narasimhan, Talluri et al. 2006). The organization, therefore, has a portfolio of products representing different life cycle stages and this makes supplier selection critical. Hence, it is the customer's interest to operate with a vendor that can provide the best set of products and consequently help the customer to optimize the product portfolio. (Narasimhan, Talluri et al. 2006).

2.3. Potential conflict of interests

The majority of the potential conflicts of interest between customer and vendor are related to timing (see TABLE 3).

TABLE 3. Potential conflicts of interests between vendor and customer during software product life cycle.

Phase	Vendor's interest	Customer's concerns
Initial development	Deliver a new product on the markets as fast as possible	Purchase a mature and reliable, fully tested software
Evolution	Offer new versions frequently	Acquire seldom and only versions with optimal functionality
	Follow the needs of the large audience	Get custom changes
Servicing	Offer patches in because large changes are expensive	Get large scale changes or a new version
	Outsource servicing	Continue vendor relationship
Phase-out	Provide no changes	Get changes for the system
Closedown	Stop providing system support	Continue the use of the system

At initial development stage, the vendor seeks to deliver a product as fast as possible while the customer is seeking for mature product with no "childhood diseases". At evolution stage, the vendor offers new versions faster than the customer necessarily is able to adopt them. At servicing stage, the customer requests for change come too late considering the life cycle stage of the software product. The correct timing of the change request would be at the stage of active evolution. The same applies with phase-out and servicing. An apparent conflict emerges if customer demands changes or support that the vendor

cannot provide no more. Legislation and contract related issues have not been included from this analysis.

3. ERP system upgrade

Software products typically evolve fast but in small steps. New versions are introduced frequently at the early stages of software product's life cycle (Mehra and Seidmann, 2006). For a consumer that has implemented a major software system, such as an ERP system, the main consideration is the question of 'when' the system has to be upgraded, no so much 'whether' it should be upgraded (Mukherji et al., 2006). Typically users do not invest in every new version but instead "leapfrog to adopting a subsequent release" (Mukherji et al., 2006). The decision for system upgrade is dependent on the system and contingent factors. Legal issues may force, lack of vendor support may push and potential competitive advantage may motivate version change.

3.1. Upgrade timing

It is important for the user organisation to understand the difference between software product's evolution and servicing (Rajlich and Bennett, 2000). The best time for consumer is to upgrade system while the system is on evolution phase because changes are then possible. If it can be predicted that servicing stage is getting closer the consumer may well start considering alternatives for survival with service patches. It is possible that the next product version has also reached the servicing stage and thus it may be necessary to consider a jump to a version or two to reach an evolvable version (if one exists).

When compared to an investment of a new system, system upgrades are less expensive and are expected to yield less benefit (Mukherji, Rajagopalan et al. 2006). However, they still are considerable investments that require organisational efforts. Frequent upgrading is costly and somewhat risky yet delaying the upgrade decision too long may "lead to loss of competitiveness" (Mukherji et al., 2006). Waiting for too long before upgrading may also increase the gap between the version in use and the version in markets in a way that later upgrading becomes difficult (Mukherji et al., 2006). Thus, it is important to determine the right moment to upgrade. The customer has to predict how the potential products will develop market-wise (Narasimhan et al., 2006), i.e. how is the product likely going to be developed and will it have product support in the predictable future.

Mukherji et al. (2006) presented a decision model for defining the optimal IT upgrade timing. They tested the model with simulations and found that "investments in upgrades are best made when the gap between new technology and current technology reaches a critical threshold. Among other factors, this threshold is influenced by technology cost, change management cost and opportunity cost." (Mukherji et al., 2006). Technology cost refers to the costs of adoption of a new version. Change management cost is the cost of upgrade deployment activities

in the organization such as the time used for learning the new routines and retraining of the users. Opportunity cost means the cost of lost opportunity. The opportunity cost can manifest itself, for instance, in a form of decreased productivity or loss of revenue due to the decision not to adopt upgraded technology or to adopt ill-fitting version.

According to the model, the objective "is to minimize the sum of this opportunity cost and the adoption cost by appropriately choosing the upgrade times [...] and selecting the levels of upgrades" (Mukherji et al., 2006). Organisation should wait until the technology cost of the new version decreases, because the newest version is the most expensive right after it has been launched on the markets. If assumed change management costs are high, then it is recommended to postpone upgrading (Mukherji et al., 2006). If a company does not invest in new technology it may lose the opportunity for higher productivity. Yet, if the new technology is incompatible with the company's existing technology the incompatibility can cause loss of revenue (Mukherji et al., 2006). Therefore, if a certain upgrade is associated with high opportunity costs then it is advisable to make the upgrade investment soon.

The results of Mukherji et al. (2006) show that the optimum time for system upgrade is when the difference between the level of technology in use and the level of the most suitable new technology level hits a critical threshold, i.e. a company has to compare the technology and change management costs to the opportunity costs and based on that comparison make the upgrading decision. "Leapfrogging" as an upgrade technique supports these findings: the most economical way of upgrading is to wait until there is a need for change, skip upgrades with only little changes, and go for the upgrade when a clearly beneficial version is available. (Mukherji et al., 2006).

3.2. Version selection

"The most suitable upgrade is defined to be the one which has the optimal combination of functionalities" and meets the needs of the adopting company and its customers (Mukherji et al., 2006), i.e. the new version does not use such technology that is incompatible with the technology the adopting company or its customers use.

The importance of a correct version selection is central in case of ERP systems. The selection of an adequate version is seen as a critical success factor for ERP system implementation (Esteves and Pastor, 2005). The version should meet the user organisation's functional requirements. It is not advisable to rush into system upgrade if the available version does not have all the needed functionality but to wait for the next version. Upgrading is a major effort and extra strain from frequent upgrading should be avoided (Esteves and Pastor, 2005). Maintenance requirements influence the version selection of an ERP system. The magnitude and frequency of mainte-

nance activities and support provided by the vendor should be assessed, as well as the phase-out timing and end of support for the particular version (Ng et al., 2003). Sometimes the newest release does not have the right functionality or only parts of needed functionality. In case like that, it is most beneficial to skip the version and adopt a follower that meets with the needs (Mukherji et al., 2006).

In this chapter we discussed the product life cycle of a major software product, like ERP. We found out that it evolves in versions that follow each other in spiral model, which are similar to the life-cycle pattern of the whole product. This should have an impact for customers upgrade decisions. In the next chapter we discuss the implications of the findings.

4. Discussion

The timing of version change is largely dependent on the product's life cycle position on the markets. New versions are available during the evolution phase of a software product's life cycle. Consequently, the upgrade decision is relevant only in software evolution phase. If the product has reached servicing or phase-out stage the opportunity of getting an adequate version upgrade is lost. That implies that the customer has to understand the life cycle of the product it uses. Also, the relationship with the vendor has to be trusting so that possible vendor's plans with respect to product development and support intentions are available and easily communicated to the customer.

When the product is in its evolution phase and the vendor responds to market needs by offering new versions, the question of system upgrade moves on to the issue of timing. According to the technology upgrade model (Mukherji et al., 2006), the best time for system upgrade is when the gap between the version in use and new available version reaches a critical threshold. This threshold can be defined based on the cost of the new version (acquisition costs), change management cost, and opportunity cost which are all influenced by customisation needs of the client. This means that the customer, on one hand, has to have a profound knowledge about the functionality of the current system and its own system needs. On the other hand, the customer has to actively follow the evolution of the product on the markets in order to be able to compare the new functionalities and changes with the existing system. A latest version may include new features that are not relevant for the customer, and thus there is no point in investing in that version. In that case, the gap itself is not enough of an indicator because particular functionality or change is required before upgrading would bring along the desired outcome. The timing of system upgrade is dependent on the life cycle phase, the existence of a suitable version and the critical threshold between the system in use and the new suitable version (see FIGURE 1).

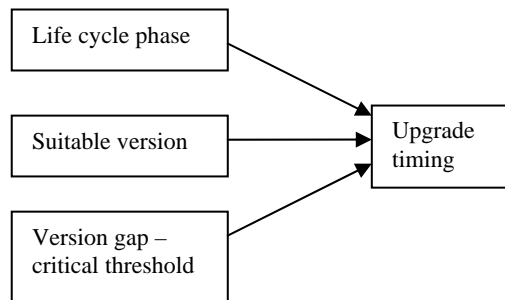


FIGURE 1. Framework for upgrade timing

Conflicting interests of vendor and customer can influence the information sharing and objective version selection. Particularly at the early stages of software product life cycle the vendor is keen to provide new versions faster than the customer needs. Towards the end of the life of the product, the setting turns upside down the customer needing more changes and support than the vendor is able to provide.

5. Summary

In this paper, the upgrading of commercial software package has been studied from the perspective of vendor and consumer's interests, and timing. Version upgrade is increasingly typical decision making point in organisations that use commercial software. Based on the previous research reports on software life cycle and IT upgrade, a framework for commercial software upgrade timing was generated. The literature analysis suggests that the timing of software product upgrading is influenced by three factors, namely the life cycle phase, the existence of a suitable version, and the gap between the system in use and the newest version.

The main reason for conflict with respect to software upgrade is related to economics. The main interest of software vendor is to sell new versions and to have the customers to follow the upgrading program in order to create revenue. The main interests of a software user organisation is to upgrade only when needed and get a version that meet the organisation's needs. These conflicting interests, from customer point of view, necessitate alertness in paying attention to the changes taking place in product life cycle and product functionality.

An interesting finding is the advantage that the customer gains from the understanding of the software product life cycle and vendor's interests respectively. As much as it has been emphasised in the past by practitioners and researchers that the vendor should know the domain of the customer in order to gain advantage over competitors, as much it seems to hold true that the customer should possess understanding of the vendor's domain in order to anticipate changes in the product life cycle. Empirical investigation of this phenomenon will be needed. Besides the timing and version gap factors, it would be interesting to study what contextual factors affect the vendor-customer relationships.

Acknowledgment

We are grateful to the Foundation for Economic Education of Finland and University of Jyväskylä for supporting this research.

References

- Cooper, R. B. et R. W. Zmud (1990). « Information technology implementation research: A technology diffusion approach », *Management Science*, 36 (2), pp. 17-31.
- Esteves, J. M. et J. A. Pastor (1999), « *An ERP Life-cycle-based Research Agenda* », First International workshop in Enterprise Management and Resource Planning: Methods, Tools and Architectures – EMRPS'99, Venice, Italy.
- Esteves, J. M. et J. A. Pastor (2005), « A Critical Success Factor's Relevance Model for SAP Implementation Projects », *Managing Business with SAP: Planning Implementation and Evaluation*. L. Lau. Hersley, PA, USA, Idea Group Publishing.
- Karsten, H. (1999), « Collaboration and Collaborative Information Technologies: A review of the evidence », *The DATA BASE for Advances in Information Systems*, 30 (2), pp. 44-65.
- Markus, M. L. et C. Tanis (2000), « The Enterprise System Experience — From Adoption to Success », *FRAMING THE DOMAINS OF IT MANAGEMENT: Projecting the Future Through the Past*. R. W. Zmud and M. F. Price (eds.), Pinnaflex Educational Resources, Inc.
- Mehra, A. et A. Seidmann (2006), « The Economics of Software Upgrades Throughout the Product Life Cycle », *39th Hawaii International Conference on System Sciences (HICSS 2006)*, Hawaii, IEEE Computer Society, Washington, DC.
- Messerschmitt, D. G. et C. Szyperski (2003), *Software Ecosystem*, The MIT Press.
- Mukherji, N., B. Rajagopalan, et al. (2006), « A decision support model for optimal timing of investments in information technology upgrades », *Decision Support Systems*, 42, pp. 1684–1696.
- Narasimhan, R., S. Talluri, et al. (2006). « Multiproduct, Multicriteria Model for Supplier Selection with Product Life-Cycle Considerations », *Decision Sciences*, 27 (4), pp. 577-603.
- Ng, C. S. P. (2001). « A Framework for Enterprise Resource Planning Maintenance and Upgrade Decisions », *The 7th American Conference on Information Systems (ACIS)*.
- Ng, C. S. P., G. Gable, et al. (2003), « A Revelatory Case Study into the Adequacy of Standard Maintenance Models in an ERP Context », *7th Pacific Asia Conference on Information Systems (PACIS)*, Adelaide, South Australia.
- Rajlich, T. V. et K. H. Bennett (2000), « A Staged Model for the Software Life Cycle », *Computer* 33 (7), pp. 66-71.
- Sawyer, S. (2001), « A Market-based Perspective on Information Systems Development », *Communications of the ACM*, 44 (11), pp. 97-102.
- Somers, T. M. et K. G. Nelson (2004), « A taxonomy of players and activities across the ERP project life cycle », *Information & Management*, 41, pp. 257–278.
- Wagner, E. L. et S. Newell (2007). « Exploring the Importance of Participation in the Post-implementation Period of an ES Project: A Neglected Area », *Journal of the Association for Information Systems*, 8 (10), pp. 480-479.
- Wei, X., C. Weiss and B. R. Nault. 2007. “Experience Information Goods: Versioning and Upgrading”, Proceedings of the 2007 INFORMS Conference on Information Systems and Technology, Seattle, Washington
- Verville, J., C. Bernadas, et al. (2005). « So you're thinking of buying an ERP? Ten critical factors for successful acquisitions », *Journal of Enterprise Information Management*, 18 (6).