A Conceptual Investigation of Maintenance Deferral and Implementation: Foundation for a Maintenance Lifecycle Model

Christopher Savage *cns993@uowmail.edu.au*
Faculty of Business, University of Wollongong
Wollongong, Australia

Karlheinz Kautz *kautz@uow.edu.au*
Faculty of Business, University of Wollongong
Wollongong, Australia

Rodney J. Clarke *rclarke@uow.edu.au*
Faculty of Business, University of Wollongong
Wollongong, Australia

Abstract

Despite the fact that society and organizations rely heavily on Information Systems (IS) and software, the maintenance of vendor-supplied IS, in particular standard package software has gained little attention within the academic literature. This paper presents a conceptual study of the current state of research concerning the reasons for deferral and performance of vendor-supplied maintenance by the purchasing organization. These reasons have so far neither been investigated together nor from that perspective. Based on a systematic literature review and taking the purchaser's viewpoint, reasons for maintenance deferral and performance are identified from the literature. They build the groundwork and foundation for a Maintenance Lifecycle and Process Model that provides a starting point to research vendor-supplied maintenance from the customer's point of view.
Keywords: maintenance deferral, maintenance lifecycle model

1. Introduction

The "dependence on critical infrastructures is increasing worldwide" ([1], p.112) and "both the impact of software on life, and our dependence on software is rapidly increasing" ([2], p.531). Companies requiring software capability that do not want to develop the capability in-house can choose to commission or outsource a unique build, or purchase the capability [3]. By purchasing from a vendor, the organization "benefits [from] generic best practices and advanced functionality supported by vendors' research capabilities" ([4], p.219). Over time, purchasing this capability has become increasingly attractive [5] and "once an organization has adopted packaged software, upgrades to newer versions are inevitable" ([3], p.153). The focus of this research is on vendor-supplied standard packaged software, abbreviated hereafter to vendor software, which is considered to be generic software, pre-created by a third-party organization for the purpose of sale or licensing. Vendor software is treated as including 3rd-party, commercial-off-the-shelf (COTS) software [5], product software [2] or packaged software [3].

IEEE defines software maintenance as "the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment" ([6], p.46) while Swanson refers to maintenance as "all modifications made to an existing application system, including enhancements and extensions" ([7], p.311). For this research we adopt a definition of maintenance both as a process and as the outcome of that process. Vendors will periodically deliver maintenance to the purchasing organization in the form of patches or upgrades ready to be applied to installed systems. The vendor develops and releases the maintenance, but each purchasing organization may have to expend significant effort to incorporate the maintenance into the production environment which may lead to the "typical option of 'doing nothing'" ([8], p.451), "IT's usual preference to 'ride [the current version] out as long as possible'" ([9], p.562) in which "neglect is the inertially easy path" ([1], p.112). This conscious or unconscious decision to postpone or delay implementation of the vendor-supplied maintenance into the operational environment is considered deferral within this paper. The study takes the purchaser's viewpoint and explores the current state of literature within the topic of maintenance deferral of vendor software. It identifies the reasons for deferral and performance of vendor-supplied maintenance by the purchasing organization. These reasons have previously not been studied from that perspective. The identified reasons build the groundwork and foundation for a Maintenance Lifecycle model that provides a starting point to research vendor-supplied maintenance from the customer's point of view.

The paper is structured as follows: some background of software maintenance and maintenance deferral is provided in the next section, followed by the description of the applied literature review method. The key themes and concepts are identified from the application of this review. A Maintenance Lifecycle Model is deduced from the literature review. A discussion and conclusion section completes the paper with a summary highlighting a key gap that provides an opportunity for further research while stating the limitations of the current research.

2. Background

Following the purchase of vendor software, the software needs support in order to maximize its operational life because "Systems are nevertheless subject to structural deterioration and obsolescence with age" ([10], p.278). By installing vendor software, purchasers will have to "be prepared for managing the impacts of [maintenance]" ([3], p.167). The cost of purchasing vendor maintenance is not a concern as many vendors employ license agreements whereby maintenance is made available without additional charge [11]. The comprehensive view on maintenance which we embrace based on Swanson [7] results in the inclusion of major and minor upgrades, patches and maintenance within this study. The maintenance period is commonly referred to as being the longest phase of the software lifecycle [5,12]. The maintenance period for vendor software begins during commissioning, as vendor-supplied maintenance is incorporated into the commissioning in order to prevent the client starting operations with an "out of date" system [13].

Within this study, deferral is treated as a conscious or unconscious decision of the purchasing organization that postpones or delays a course of action. Implicit within this definition of deferral is that the postponed action will have to be performed at some future time. Referring to road maintenance, Harvey ([14] p.34) captures the essence of maintenance deferral in any realm as "deferring maintenance can be seen as a form of borrowing. Funds are saved in the short-term at the expense of higher outlays in the future."

Deferral becomes a critical issue for the purchaser of vendor software when the vendor declares an "end of life" (EOL) date, indicating that further maintenance ceases for this version [15]. This forces the business to accept a new risk of using a component of unsupported IT infrastructure or software, or perform maintenance to move onto a supported version of software [9]. In reviewing papers relating to the deferral of vendor-supplied maintenance, this paper will investigate how EOL can become a problem for purchasers of vendor software due to the purchaser repeatedly deferring the adoption of newer versions of the software. The backlog of software maintenance activities for software packages creates a poorly-understood risk for organizations [13] and warrants further research.

3. Review Method

Our conceptual investigation is based on a systematic literature review. The execution of the review and the structure of reporting its results follow a concept-centric literature review approach [16] to ensure that repeatable data gathering and logical analysis support the discussions presented and conclusions drawn. Kitchenham and her associates [17,18] provided guidance for a systematic review. The review progresses through a sequence of filtering results as presented by [19]. The addition of a preliminary, informal step expands the number of initially considered papers, thereby reducing the risk of accidently eliminating papers during the initial search [18].

In preparation for the systematic review, an unstructured review of publically available literature through a State Library was conducted using the terms "maintenance deferral", "project prioritization" and "project prioritisation". Iterative snowball addition of key words and concepts from the resulting papers created 10 different search terms related to the core concepts listed above. To maximize the scope of literature, the initial search was not limited to topic-specific databases, popular publications or peer-reviewed papers. This is consistent with the advice that a wide net should be cast in order to consider all published articles in a field [16].

The Web of Science^{TM} database was selected for this review due to the wide cross-discipline nature of the index including the Association for Information Systems' Senior Scholars "basket of 8" journals and all but two journals from The Financial Times 45 top journal list. For this review, the titles of about 14900 papers were evaluated. Through the different screening processes a total of 40 papers were included into the review. The results of this analysis are presented in the next section.

4. Results

Despite the broad search terms, every item that passed the critical review referred to the maintenance deferral problem from a vendor's perspective; no papers addressed the problem directly from the purchaser's perspective. The papers were published over a period of nearly 40 years with the first one appearing in 1977. This demonstrates that the topic of maintenance deferral is not new. The geographical distribution of papers indicates that the deferral problem discussed here provides a "Western" view of the issue and may not be globally generalizable. The literature review filtering criteria of English-language papers will have influenced this distribution. Five concepts and themes emerged from the literature; they are: (1) maintenance of vendor software is a problem, (2) there is too little research on the topic, (3) reasons for maintenance deferral do exist, (4) deferral has consequences (5) there are reasons for maintenance implementation.

4.1 Maintenance of Vendor Software is a Problem

The literature acknowledges that adoption of vendor software causes a maintenance problem for the purchasing organization [1,8,9,13,20]. No paper expressed a dissenting opinion that vendor software is free of maintenance impacts. Within this acknowledgement, several reasons which we will discuss in the following subsections were identified that led to organizational caution when assessing vendor-supplied maintenance before implementing it into production environments.

4.2 There is too little Research on the Topic

The literature includes numerous calls for further investigation into the maintenance of vendor-supplied systems as well as for the maintenance deferral phenomenon and highlights the increasing issue of maintenance backlog in IT systems and infrastructure.

Already 1983 Lientz ([21], p.277) requested "much more research is needed in maintenance". In 1995 Swanson stated "I wouldn't do the same [1979 Dimensions of Software Maintenance] study [today]. … I would try to focus on the maintenance of commercial software packages ... Or, I would address maintenance from the user perspective, which has been largely ignored." ([7], p.307). In the same vein several authors [3,9,20,22] lament the neglect of investigations into vendor software maintenance. Khoo et al. ([22], p.334) implicitly call for more research stating "Although our research provides an initial investigation into the phenomenon of support upgrades, the empirical support for our findings were limited to a single upgrade case."

Hybertson et al. ([23], p.215) similarly state that "COTS use is increasing, and maintenance issues of COTS-intensive systems need to be articulated and addressed." They are supported by Reifer et al. ([15] p.95, 96) who say "Currently, few COTS software lifecycle models address [Component-Based System] maintenance processes" and demand "To make better decisions relative to [Component-Based Systems], we need empirical knowledge. To gain this knowledge, we must understand more fully the lifecycle processes people use when harnessing COTS packages." The absence of academic framework(s) or in-depth research addressing the organizational behavior during the period between the vendor publishing maintenance to the purchaser and the tipping point that triggers the maintenance to be applied to the purchaser's system(s) is also stated in [3].

Literature relating to the initial investment decision and deriving the full expected benefits from a past investment decision were prevalent, an observation supported by [22]; however software maintenance is either mostly ignored both by research and practice [24] or is simply not attractive, considered "less glorious" [25] and suffering a negative image with developers and managers involved in the process [26-28].

Finally, only few papers did employ theoretical models to describe some aspects of the vendor-supplied maintenance deferral issues. Communicative framing theory is used to show how consistent messages and actions prepared and supported users through the application of a major IS maintenance activity through the use of a galvanizing negatively-framed message [22]. An inductive research strategy and comparative analysis is presented in [9] to construct a theoretical model about the interaction of factors influencing upgrade decisions by adopting a critical realism approach to explain motives, contingencies and dependencies impacting the decisions. Finally, Khoo, et al. [3] extend Swanson and Beath's [25] Relational Foundation model to incorporate the vendor relationships in an explanation of the impacts that vendor software upgrades have on business and IS stakeholders. Hanna and Martin [29] discuss a model that incorporates vendor-supplied maintenance into a larger Repair Level Analysis, but complain that IS researchers and practitioners have so far failed to embrace such modeling within pure IT systems. In summary, there is too little research into the maintenance of vendor-supplied systems.

4.3 Reasons for Maintenance Deferral Do Exist

From the literature, a common theme of reasons for maintenance deferral can be deduced. In almost all cases, analysis of these reasons often expressed as risks suggests that their consequences can be avoided through the deferral of vendor-supplied maintenance, or exercising the 'doing nothing' option. Table 1 presents the reasons for deferral of maintenance expressed across literature assessed for this conceptual study.

The risk of losing customizations, configurations, or interfaces was most recognizable in the literature. It extends beyond the technology-based concerns into the realm of the user as "Users also create idiosyncratic adaptations and workarounds to overcome limitations in any customized software" ([22], p.329) that could be impacted through the application of maintenance.

Almost as prevalent was the risk that vendor-supplied maintenance would have a huge cost associated with it [13]. As purchasing organizations implement vendor-supplied systems to gain a commercial advantage [20] any planned or unplanned expense in monetary or effort-based terms may detract from this profit-making goal. In some of the few direct references to deferral, cuts and limits in maintenance budgets are a common occurrence and the flow-on deferral of maintenance is a direct result [15,23]. A more general economic downturn may also lead to maintenance being seen as too costly [30].

Table 1. Reasons for maintenance deferral

Loss of customizations, configurations, interfaces	Complicated & expensive test environments & infrastructure
Huge costs	Disrupting to the organization & productivity
Chain reaction of cascading maintenance	Unforeseen impacts, impossible to complete tests
Training efforts and steep user learning curve	Dependence on vendor claims of suitability
Poor quality, conflict for existing & new IS resources	Dependence on vendor documentation
Effort to analyze, test, or implement	Conflict with the vendor
Inconvenient rate & time of arrival	Resistance & user revolt
Disturbing the existing IS equilibrium	Additional work for expert users to train others
Difficult or complex	Requiring a re-certification for a certified system

The risk that maintenance to one system will cause a chain reaction of integration updates and backward-compatibility issues has been very common in the literature. Both minor inconveniences such as missing device drivers following operating system maintenance requiring replacement of printers, faxes and scanners [3] and a case of thirteen linked vendor-supplied systems requiring upgrade [31] related to this risk. The risk of cascading maintenance applies also to internal maintenance requirements of a vendor-supplied system. A mandatory maintenance action on one module may cause issues requiring further maintenance of a separate module [13].

A further reason that appears regularly in the literature is related to training effort and user's learning curves. Khoo et al. ([22], p.332) state "Because SAP upgrades usually involved downtime and training, business users normally preferred to defer an upgrade as long as possible." This is supported by [9,13].

The risk that a vendor-supplied maintenance release will be of poor quality and introduce bugs and conflicts between existing and new IS resources appears already in work of the early 1980ies [32] and is frequently confirmed [7,22].

The risk that a vendor-supplied maintenance release will consume a tremendous amount of effort to analyze, test, or implement is significant and justified. The need for testing is not eliminated through the implementation of vendor software [33] and the literature reports testing and implementation efforts between six months and a year for some upgrades [8,22].

The unpredictable behavior of vendors, where "it is difficult to determine when the software will be released" ([2] p.533), or simply the "burdensome … rate of change" for vendor software ([5], p.362) leads to risks concerning the inconvenient rate and time of arrival of maintenance releases.

A number of studies included the risk that maintenance will also disturb the existing equilibrium of information systems in the organization [1,7,20,31]. A risk that a maintenance project can be as difficult and complex as the original installation is also existing [1,3,5,20,23].

Also related to unpredictable behavior of vendors is a risk of the unforeseen, related to the impossibility of fully testing a maintenance release and its un-assessable impacts and side-effects [2]. The risk of the unknown is implicitly common to all risks listed here, but there is a specific risk of the unforeseen, that even when everything is assessed and mitigated, something might go wrong with a concrete example of this unforeseen risk being pointed to as "Unexpected problems with file sharing in Access" in ([7], p. 164).

An example of the risk of maintenance disrupting the organization and its productivity was the failure of a new feature in upgraded software causing "a mess for about three weeks" ([3], p.161). A second example of organizational disruption [22] saw a company undergoing slowdown in performance and system lockouts subsequent to a three-day outage to implement the upgrade and in another case "files missing" ([3], p.162) as a side-effect from an upgrade.

Some papers lamented the complications and costs of maintaining environments for testing [5,23,32] as a specific risk with one recording three separate environments, development, test, and production, in an organization in order to manage and maintain their vendor-supplied system [13].

Because organizations need the vendor for the maintenance of vendor software, they must rely on the vendor claims concerning the suitability of the maintenance release; a risk pointed out by [5,12,31,34,35]. A similar requirement of dependence and a related risk is specifically valid for the documentation that accompanies a release as such documentation "might be incorrect on incomplete" ([12], p.13).

Inevitably, applying maintenance to an operational system may cause conflict with the vendor as illustrated by [7]: "During the … testing phase, [Information Systems] staff identified many problems that they attributed to [the] software, but the vendor countered that the problems were related to client [organization] configuration decisions" ([7], p.165). A risk of conflict with the vendor can be deduced from this and has been confirmed by [31,34,36]. A risk of maintencance leading to resistance and user revolt caused by changed software is also identified [3,22]. Additional work for expert users in the form of training other employees is another reason for maintence deferral [3]. Finally, a risk that upgraded software might require a re-certification for a certified system has been stated by [5].

4.4 Deferral has Consequences

Deferral can be a logical, considered course of action when the risk of implementing the maintenance is calculated to be unacceptable [5]. An example of unacceptable risk is an incompatibility between the maintenance item and its environment, vendor disclosed [35] or otherwise identified or an identified threat to the stability of a system associated with a major release [9].

The consequences of maintenance deferral can otherwise be to avoid expense in the short term, however the legitimacy and suitability of this approach assume that no trigger event will occur. Should a trigger event occur and be ignored, possible consequences include economic damage to the company [38], higher expenditure and forced outages at a later time [30], or even demise of the purchasing organization itself [5].

Although IT maintenance can be deferred for one to two years, extended periods of deferral can lead to "the application portfolio risks getting dangerously out of date and a systemic risk" ([39], p.1). The risks of systemic failure create a situation of positive-feedback where "the more different infrastructures that fail concurrently, the more difficult it becomes to restore service in any of them" ([1], p.112).

For organizations that have an understanding of the actual state of their systems, the act of maintenance deferral can be a considered action to save expense, and improve stability leading into a system retirement or replacement "as the end of any system's life is eventually foreseen, the maintenance effort itself may be moderated" ([10], p.279).

One possible consequence following repeated deferral is to completely separate from the vendor's support model and to "go it alone" through either maintaining the system in-house, or paying for bespoke support, possibly receiving a lower priority than up-to-date clients of the vendor [22]. However, the approach of deferring maintenance becomes precarious when vendor-supplied maintenance "that we require urgently" arrives, but has a dependency on a "backlog" of un-installed changes, which occurs because the vendor "seems to assume that you are up to date" ([13], p.100).

4.5 There are Triggers for Maintenance

Mukherji et al. [40] put forward that investments in upgrades are best made when the gap between the new technology and the one currently in use reaches a critical threshold. A theme of identifiable trigger events, which cause this threshold to be reached immediately preceding the implementation of vendor-supplied maintenance, emerges from the literature. Table 2 summarizes the identified triggers and reasons for maintenance implementation.

Table 2. Triggers and reasons for maintenance implementation

Need for increased business benefit	Standardization resulting from acquisition or merger
Avoid EOL date when vendor supports stops	Remain current with the marketplace
React on changed hardware requirements	Respond to a massive social change or innovation
Resolve an error relevant to the purchaser	Change to such environment such as legislation
Satisfy company policy	React to release of vendor maintenance
Standardization to remain compatible with external parties	Eliminate or contain a security threat

Satisfying the need for increased business benefit is the most reoccurring theme within the literature concerning triggering the implementation of maintenance. This could be achieved through new functionality and features available within a newer release that fulfills user requests or requirements especially for improved performance. In this context, an aspiration of first-mover advantage also spurs maintenance [40]. Vendors declaring an EOL date for support of a particular version were an often referenced trigger for maintenance implementation [20]. The adoption of vendor software creates a lock-in situation where the purchasing organizations become dependent on the software vendor to provide them with software functionality and technical support [9]. This means that a vendor declaring an end to that support represents a significant risk to the purchasing organization. Lientz and Swanson [32] in their early studies on maintenance identified a non-software trigger event, the requirement to move from obsolescent hardware or to upgrade the hardware platform to mitigate hardware availability and support issues. This has been confirmed through the following years [31]. Several studies [10,13,20,22,23,41] identify the resolution of an error relevant to the purchaser as a further trigger for maintenance. Policy within the organization aims to assist with determining the occurrence of a trigger event, however apparently contradictory policies with the same aim were identified in separate studies: one policy required a company remain within vendor-support version requirements [8], another one requested to upgrade every one and a half years [22]. A rationale for standardization as another trigger for maintenance is the need to remain compatible with external parties that interface an organization's information systems [37]. A need to standardize IS infrastructure resulting from a business acquisition or merger may also trigger maintenance [8].

A regular trigger for maintenance is the need to remain current with the marketplace [20,21,33,40] as is change to the external environment such as legislation [21] to be dealt with legal-change-patches [20]. In addition, other environmental factors such as competitive pressure and general social and cultural factors were stated as triggers [21]. Major social change was also identified as triggering maintenance: e.g. the introduction of the Euro currency within the European Union [41]. Some papers alluded to the vendor maintenance release as a simple and sufficient trigger for a client organization's reaction to implement it [4,13,20,33]. Finally, exploits or threats that increase the risk in a safety-critical, life-critical or secure system are possible triggers for maintenance [5,38].

5. Deduction of a Maintenance Lifecycle

Beyond the identification of the reasons for maintenance deferral and implementation our literature review also uncovered concepts which allow us to put forward a maintenance lifecycle model. IEEE [6] puts forward a software lifecycle consisting of 8 phases, one of them being the operation and maintenance phase. It is defined as "the period of time in the software lifecycle during which a software product is employed in its operational environment, monitored for satisfactory performance, and modified as necessary to correct problems or to respond to changing requirements" ([6], p.52). Through the synthesis of concepts spanning multiple critically reviewed papers, we deduce and propose a dedicated maintenance lifecycle from ideas not previously unified. This cycle begins with acquisition of the asset that creates a need to maintain the investment [1]; a trigger event causes maintenance to be required [5,9] ; the maintenance activity is planned [31]; the purchasing organization's IS and software users are prepared for the maintenance [22]; the maintenance is implemented; and the implications to the organization arising from the maintenance are stabilized [3]. Figure 1 uses the Software Lifecycle to demonstrate the placement of this maintenance lifecycle.

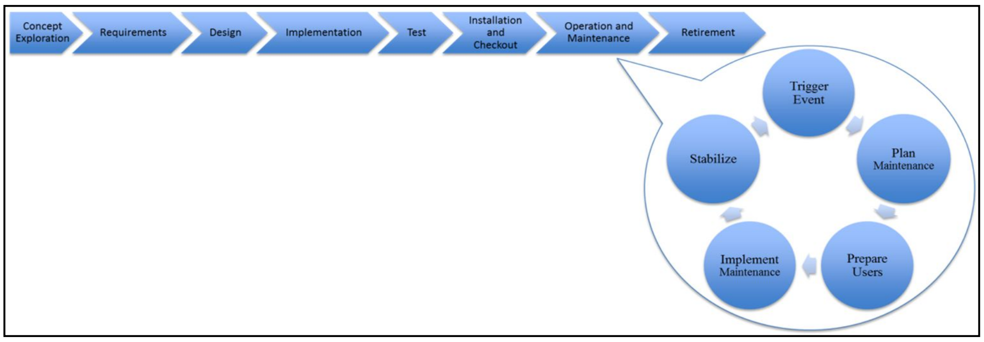

Fig. 18. A Maintenance Lifecycle model

The proposed lifecycle is repetitive which refers to ongoing enhancements following the initial system commissioning and stabilization [21]. It is also reminiscent of Deming's Plan-Do-Act-Check cycle, in that it iterates through the states. The difference within the maintenance cycle is an explicit "wait" state before a trigger event, while the need for the next planning phase arises. In other words, there is not necessarily an automatic progression prior to the next trigger event.

6. Discussion

This study provides a summary of the reasons for maintenance deferral and implementation. It thereby advances three research problems stated by Gable et al. [34] in an early research framework for large packaged application software maintenance: (1) In exploring the rationale for deferral, the identified reasons refer to the drivers for a maintenance decision. (2) It takes up the question of to what extent can maintenance can be avoided through packaged software solutions? Implicit in this question is the assumption that maintenance can be avoided, which is addressed by the deferral aspect within this study. (3) Lastly, a maintenance lifecycle model is deduced as a generic concept across all possible vendor-supplied systems and demonstrates that packaged software maintenance concepts are in fact generic and extensible beyond a particular vendor's product.

The study had to address several challenges. The first challenge was gaining a suitable view of the concept of software. For as long as IT, IS, and software investments have existed, there have been attempts to classify the artifacts in a way common to other investments. Through the adoption of a technical vendor viewpoint they can be divided into infrastructure, tools or applications [2]. An alternative view is to understand IT, IS and software as representing an asset [41]. This treatment of software as an asset supports references within the study to deferral behaviors within the engineering realm and its physical assets. The next hurdle was the very definition of the key term maintenance. The reviewed papers provided conflicting definitions of maintenance, patches and upgrades that confused attempts to synthesize a clear picture of the topic. This was reinforced through the diverse search terms required to capture the documents assessed. Therefore the IEEE [6] definition of maintenance as a process and Swanson's [7] view of maintenance as an outcome or product were adopted. Although vendor-supplied software maintenance can add new functionality, the customer judges the maintenance as required in order for the software asset to remain useful. The study is then based on an understanding that the maintenance phase begins following the purchase of vendor software, not its activation. This is because the first maintenance releases "will occur before the system's initial delivery" ([33], p.53).

Deferral has negative connotations as "deferred maintenance is an exercise that often detracts from the more fundamental task of attacking the problem itself. Because of the implication that deferral has been caused by neglect and not by conscious planning, administrators shy away from approaching the main job" ([42], p.43). This study demonstrates that deferral has both legitimate and neglect-based causes. From the first origins of maintenance with the creation of tools and structures, items were used until they failed [43]. Through the industrial revolution, systems became more complex but maintenance, apart from routine lubrication, remained largely something performed at the point of failure. During World War II the need for operational fighter aircrafts created a need for preventative maintenance, maintenance before failure, therefore creating a function to support availability. From the results of this study, the need for a trigger event before implementing maintenance strongly indicates that some IS owners are still behaving in a mode of operating-to-failure or, operating-to-obsolesce their software investments.

The need for different approaches to internal processes caused by the move from a traditional in-house development team to vendor software is poorly understood and provides a lens to understand the vendor-supplied maintenance deferral question [33]. It is possible that some purchasing organizations fail to make the change to utilizing packaged software based processes and therefore remain unaware of the pervasive ramifications both to people and processes that are triggered by implementing a vendor-supplied product and subsequent maintenance. A further area where businesses may not fully appreciate the complexity of vendor-supplied maintenance is that traditional methods of costing, cost-benefit analysis (CBA), return on investment (ROI) and risk-analysis, do not translate well from an in-house to a vendor-supplied environment unless specific allowance is made for vendor-supplied maintenance activities. Although a maintenance release may have no compelling reason to be implemented, for example, a low ROI, a negative CBA, no improvement to risk profile, a later more critical maintenance item may have a dependency on the earlier one. If the risk of deferral is not factored into the original implementation decision, the cost and time required to utilize the later critical maintenance release will be under-appreciated.

Traditional budgeting sets an IT department's operating budget on an annual basis. Translation of this budget into staffing allocation extends the assumption of fixed budget into an assumption of staff costs and staff as fixed input, available to perform work. Contained within this work is the effort required to analyze, test, and implement vendor-supplied maintenance into the production environment. However, this study has shown that vendor behavior does not always support such a predictable cycle of resource and budget availability. A case study [20] emphasized that if mandatory, maintenance were implemented when it arrived from the vendor, and 80% of the annual maintenance effort would be consumed through fortnightly implementations; batching updates into larger, less frequent implementation activities significantly benefited the reduction of total effort required. This is an example of planned and managed maintenance deferral. An implication of this somewhat random vendor behavior of maintenance production is to introduce a variable requirement for maintenance work into an organization that is geared for a static level of effort. Incorrectly accounting for this variability through the budgeting cycle could introduce a financial constraint on the ability to implement vendor-supplied maintenance. The proposed lifecycle model with its planning and preparation activities might support this processes.

When surrendering control of maintenance to the vendor, an organization largely relinquishes the ability to manage or dictate the content of an individual maintenance package and thus either might defer as long as possible or see no need for maintenance. Organizations may assume that once purchased no allocation of time or effort is required for ongoing maintenance, and that the problem was solved in the original purchase. This view is reinforced by purchase decisions that fail to build the operational and maintenance costs into the decision process. An alternative outcome from the trigger event may be to re-assess the vendor software and determine that a replacement is necessary as presented in a case for optimally timed system replacement in response to this outcome [44]. In another case an examination of the issues lead to a system being retired, again referencing the maintenance cost/effort of the system to support the decision [10]. In this case, a valid approach is to operate the current system without further maintenance. This is an example of conscious deferral. It is addressed in the proposed maintenance lifecycle wait state with a subsequent trigger event, which results in leaving the maintenance cycle.

7. Conclusions

This study shows, and this is in particular emphasized in work concerning maintenance in security and safety sensitive environments [38], that maintenance behavior in purchasing organizations is not universal, but can be unified into a maintenance lifecycle model that takes different contexts into account. It presents a comprehensive review of the reasons for maintenance deferral and implementation within the area of vendor software from the purchaser's perspective. As such it provides a solid foundation for further attention and research in this long neglected area and demonstrates that execution of a broad systematic literature review relating to a sparsely published area of research informs research through the deduction of themes and concepts as well as a foundational lifecycle model from an expansive selection of literature.

This study supports practice with an understanding of organizational behavior with regard to maintenance deferral and implementation. Through an awareness of common reasons, it can help identify deferral causes, develop responses to these causes and forecast the upcoming need for and implementation of maintenance within an organization. The IS community can advance this process through further research, resulting in theories, frameworks and models that assist practice in navigating the deferral problem in the future. The derived reasons have been distilled from work where they form sometimes an incidental mention during the study of other topics. Though important enough to warrant mentions, the list cannot be considered complete without further empirical testing. Further research is thus needed to validate the identified concepts and themes as well as the Maintenance Lifecycle model and its usefulness for understanding and resolving the maintenance deferral problem with empirical case studies. The differences between reasons leading to the deferral of maintenance and reasons leading to the implementation of maintenance show that understanding the motivations that require an upgrade decision in the present, do not explain all motivations for deferral in the past. Thus a conceptualization of deferral as a process that recognizes that deferment and implementation of maintenance take place in a complex social process may be beneficial in such investigations.

Any study has its limitations. The systematic literature review was performed by the first author without a peer-review and dispute resolution process for the evaluation of each paper against the filtering criteria. The review was limited to papers published in English. Restricting the search to the Web of Science^{TM} exposed the review to constraints of the data source concerning publishing dates, use of search operators and keywords. These limitations may exclude some valid articles. This study has focused on the organizational behaviors relating to single vendor-supplied systems; organizations can operate with multiple, integrated vendor software packages that increase the complexity of any maintenance decision [12]. This has also to be taken into account in future research.

References

1. Horning J., Neumann, P.G.: Risks of Neglecting Infrastructure, Communications of the ACM (51:6), pp. 112-112 (2008)
2. Xu, L., Brinkkemper, S.: Concepts of Product Software, European Journal of Information Systems (16:5), pp. 531-541 (2007)
3. Khoo, H.M., Robey, D., Rao, S.V.: An exploratory study of the impacts of upgrading packaged software: a stakeholder perspective, Journal of Information Technology (26:3), pp. 153-169 (2011)
4. Maheshwari, B., Hajnal, C.: Total systems flexibility and vendor developed software: Exploring the challenges of a divided software life cycle, IEEE International Engineering Management Conference 2002, Vols I&II, Proceedings: Managing Technology for the New Economy, IEEE, New York, pp. 218-223 (2002)

5. Carney, D.,Hissam, S.A., Plakosh, D.: Complex COTS-based software systems: practical steps for their maintenance, Journal of Software Maintenance-Research and Practice (12:6), 2000, pp. 357-376 (2000)
6. IEEE: IEEE Standard Glossary of Software Engineering Terminology, pp. 1-84 (1990)
7. Swanson, E.B. Chapin, N.: Interview with Swanson, E. Burton, Journal of Software Maintenance-Research and Practice (7:5), 1995, pp. 303-315 (1995)
8. Ng, C.S.P.: A decision framework for enterprise resource planning maintenance and upgrade: A client perspective, Journal of Software Maintenance and Evolution - Research and Practice (13:3), 2001, pp. 431-468 (2001)
9. Khoo, H.M., Robey, D.: Deciding to upgrade packaged software: a comparative case study of motives, contingencies and dependencies", European Journal of Information Systems (16:5), pp. 555-567 (2007)
10. Swanson, E.B., Dans, E.: System life expectancy and the maintenance effort: Exploring their equilibration, MIS Quarterly (24:2), pp. 277-297 (2000)
11. Cusumano, M.A.: Changing software business: Moving from products to services, Computer (41:1), pp. 20-27 (2008)
12. Vigder, M.R., Kark, A.W.: Maintaining COTS-based systems: Start with the design, in Fifth International Conference on Commercial-off-the-Shelf, IEEE Computer Soc, Los Alamitos, 2006, pp. 11-18 (2006)
13. Ng, C.S.P., Gable, G.G., Chan, T. Z.: An ERP-client benefit-oriented maintenance taxonomy, Journal of Systems and Software (64:2), 2002, pp. 87-109 (2002)
14. Harvey, M.: Optimising road maintenance, in International Transport Forum, OECD/ITF, Paris, 25-26 October 2012, viewed 24 June 2013, <http:// www.inter nationaltransportforum.org/jtrc/DiscussionPapers/jtrcpapers.html (2012)
15. Reifer, D.J., Basili, V.R., Boehm, B.W., Clark, B.: Eight lessons learned during COTS-based systems maintenance, IEEE Software (20:5), 2003, pp. 94-96 (2003)
16. Webster, J., Watson, R. T.: Analyzing the Past to Prepare for the Future: Writing a Literature Review, MIS Quarterly (26:2), pp. 13-23 (2002)
17. Kitchenham, B., Brereton, O.P., Budgen, D.,Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering - A systematic literature review, Information and Software Technology (51:1), pp. 7-15 (2009)
18. Kitchenham, B., Brereton, O.P.: A systematic Review of systematic Review Process Research in Software Engineering, Information and Software Technology (55:12), pp. 2049-2075 (2013)
19. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review, Information and Software Technology (50:9-10), pp. 833-859 (2008)
20. Ng, C.S.P., Chan, T.Z., Gable, G.G.: A client-benefits oriented taxonomy of ERP maintenance, in IEEE International Conference on Software Maintenance, Proceedings: Systems and Software Evolution in the Era of the Internet, IEEE Computer Soc, Los Alamitos, 2001, pp. 528-537 (2001)
21. Lientz, B.P.: Issues in Software Maintenance, Computing Surveys (15:3), pp. 271-278 (1983)
22. Khoo, H.M., Chua, C.E.H., Robey, D.: How organizations motivate users to participate in support upgrades of customized packaged software, Information & Management (48:8), pp. 328-335 (2011)
23. Hybertson, D.W., Ta, A.D., Thomas, W.M.: Maintenance of COTS-intensive software systems, Journal of Software Maintenance-Research and Practice (9:4), pp. 203-216 (1997)
24. Ketler, K., Turban, E.: Productivity Improvements in Software Maintenance, International Journal of Information Management (12:1), pp. 70-82 (1992)
25. Swanson, E.B., Beath, C.M.: Maintaining Information Systems in Organizations, John Wiley & Sons, New York, (1989)
26. Biskup, H., Kautz, K.: Maintenance: Nothing Else but Evolution?!, Information Technology & People (6:4), pp.215 – 231 (1992)

27. Tan, W.G., Gable, G.G.: Attitudes of maintenance personnel towards maintenance work: A comparative analysis, Journal of Software Maintenance-Research and Practice (10:1), pp. 59-74 (1998)
28. Junio, M.G.A., Malta, N.N., Mossri, H.D., Marques-Neto, H.T., Valente, M.T.: On the Benefits of Planning and Grouping Software Maintenance Requests,15th European Conference on Software Maintenance and Reengineering (CSMR), pp. 55-64 (2011)
29. Hanna, M.L., Martin, L.: Quantitative determination of maintenance concepts for COTS based systems, in Annual Reliability and Maintainability Symposium, Proceedings, IEEE, New York, 2007, pp. 478-481 (2007)
30. Bloch, H.P.: Deferred Maintenance Increases Pump Failures, Power (155:2), p. 16 (2011)
31. Anderson, W., McAuley, J.: Commercial off-the-shelf product management lessons learned - Satellite ground control system (SGCS) upgrade, in Fifth International Conference on Commercial-off-the-Shelf, IEEE Computer Soc, Los Alamitos, pp. 206-213 (2006)
32. Lientz, B.P., Swanson, E.B.: Problems in Application Software Maintenance, Communications of the ACM (24:11), pp. 763-769 (1981)
33. Brownsword, L., Oberndorf, T. Sledge, C.A.: Developing new processes for COTS-based systems, IEEE Software (17:4), pp. 48-55 (2000)
34. Gable, G.G., Chan, T.Z., Tan, W.G.: Large packaged application software maintenance: a research framework, Journal of Software Maintenance and Evolution-Research and Practice (13:6), pp. 351-371 (2001)
35. Bachwani, R., Crameri, O., Bianchini, R., Zwaenepoel, W.: Recommending software upgrades with Mojave, Journal of Systems and Software (96), pp. 10-23 (2014)
36. Arora, A., Krishnan, R., Telang, R., Yang, Y.B.: An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure, Information Systems Research, (21:1), pp. 115-132 (2010)
37. Ellison, G., Fudenberg, D.: The neo-Luddite's lament: excessive upgrades in the software industry, Rand Journal of Economics (31:2), pp. 253-272 (200)
38. Arora, A., Forman, C., Nandkumar, A., Telang, R.: Competition and patching of security vulnerabilities: An empirical analysis, Information Economics and Policy (22:2), pp. 164-177 (2010)
39. Gartner: Gartner Estimates Global 'IT Debt' to Be $500 Billion This Year, with Potential to Grow to $1 Trillion by 2015, No. 1, Gartner, gartner.com, p. 1 (2010)
40. Mukherji, N., Rajagopalan, B., Tanniru, M.: A decision support model for optimal timing of investments in information technology upgrades, Decision Support Systems (42:3), pp. 1684-1696 (2006)
41. Ben-Menachem, M.: Towards management of software as assets: A literature review with additional sources, Information and Software Technology (50:4), pp. 241-258 (2008)
42. Kaiser, H.H.: Deferred Maintenance, New Directions for Higher Education (30), pp. 41-54 (1980)
43. Visser, J.K.: Maintenance management - A neglected dimension of engineering management, in IEEE Africon, Vols 1 and 2: Electrotechnological Services for Africa, IEEE, New York, pp. 479-484 (2002)
44. Tan, Y., Mookerjee, V.S.: Comparing uniform and flexible policies for software maintenance and replacement, IEEE Transactions on Software Engineering (31:3), pp. 238-255 (2005)