IT Innovation Squeeze: Propositions and a Methodology for Deciding to Continue or Decommission Legacy Systems

G. Gangadharan, Eleonora Kuiper, Marijn Janssen, Paul Luttighuis

IT Innovation Squeeze: Propositions and a Methodology for Deciding to Continue or Decommission Legacy Systems

G.R. Gangadharan[1], Eleonora J. Kuiper[2], Marijn Janssen[3], and Paul Oude Luttighuis[4]
1 IDRBT, Hyderabad, India
grgangadharan@idrbt.ac.in
2 Belastingdienst, Apeldoorn, Netherlands
ej.kuiper@belastingdienst.nl
3 Delft University of Technology, Delft, Netherlands
M.F.W.H.A.Janssen@tudelft.nl
4 Novay, Enschede, Netherlands
paul.oudeluttighuis@novay.nl

Abstract. Organizations have been confronted with fast moving developments in the Information Technology (IT) sector over the past decades. Many new technological paradigms have emerged and left a landscape of legacy in which more and more money is spend on maintaining this landscape at the expense of innovating. Especially where business requirements put time pressure on the evolution of the IT landscape the decision whether to continue and maintain legacy systems or to decommission legacy systems in time has become a huge challenge. We formulate a set of propositions influencing the decision to decommission or continue legacy systems. This set of propositions is derived from literature and interviews with high level managers of organizations. Software characteristics, development methods, dependency of systems, lock-in, system complexity, new technologies and system ownership influence the decision whether to decommission or to maintain a system. We conclude this paper by proposing a methodology that helps organizations in finding the right balance between discontinuing and maintaining legacy systems.

1 Introduction

As systems are more and more connected with each other, both inside and outside organizations, they become more dependent on each other. The more dependencies, the more difficult these systems can be replaced. Application portfolio management is a means to manage the system landscape (Jeffery and Leliveld, 2004, McFarlan, 1981, Hamilton, 1999).Information technology portfolio management is the management of IT as a portfolio of assets similar to a financial portfolio (Jeffery and Leliveld, 2004). In a portfolio management approach decisions on whether to invest in IT and to decommission systems are made. Portfolios often contain an overview of existing systems and the budget spend on it. The control and

maintenance of these existing legacy systems take away a significant and increasing portion of the total information technology (IT) budget. There is a continuous increase in spending on controlling and maintaining legacy systems since the 1960s and organizations are focusing less on IT innovation. All activities not related to the control and maintenance of the existing IT landscape are referred to as IT innovation in this paper. As such we take a very broad view on IT innovation, as it includes all activities related to the change of the existing IT-landscape. Our focus is on budget, but also without any budget innovation can be accomplished. Although we acknowledge this, this factor is not taken into account in this study. IT innovation can cover a broad range of activities and can take many forms including making business processes into automated IT functions, developing applications that open new markets, or implementing desktop virtualization. Some organizations spend up to 80% of their IT budget on keeping legacy systems running, leaving only 20% for innovation (Glass, 2006). If this trend is not reversed there might be hardly any budget available for innovating and developing new IT. This phenomenon is termed the *IT innovation squeeze*, since innovation is squeezed out by the existing IT landscape (Gangadharan, Kuiper and Oude Luttighuis, 2010).

According to Hillenius (2006) a pension company spent about 75% of its IT budget on maintenance as follows. Half of the total amount was spent on maintenance of the hardware and network infrastructure. Keeping applications running took another quarter. New projects and adapting the IT to new rules and regulations required the remaining quarter. A United States university reported that the cost of software maintenance of its ERP system was increasing 50% per year and the maintenance costs of its CISCO systems increased 12% to 15% per year (Lowe, 2009). Jones (2007) estimated the yearly maintenance cost prospects for software projects for the first five years of maintenance ahead, related to the development costs. He reported an average 860 USD maintenance costs per 1,000 USD development costs per year, as well as a worst case, in which 1,116 USD maintenance costs were required per 1,200 USD development costs per year. The maintenance costs increased at a rate of 20% per year, yielding ever worse figures when the time horizon extended beyond five years. Other studies demonstrated the increase of the relative costs of software maintenance over the years. See Table 1 for figures over the years 1980 – 2000.

Table 1. Gradual increase of relative software maintenance costs.

Study	**Relative software maintenance costs**
Erikh (2000)	> 90%
Eastwood (1993)	75%
Port (1988)	60-70%
Lientz and Swanson (1980)	> 50%

All these figures indicate a tendency that over the years an increasing percentage of budgets is spent on controlling and maintaining existing IT. So far we have not

found organizations in literature or in practice with a decreasing percentage of budget, spent on existing IT. If the maintenance activity of existing systems continues to outpace new software development on contemporary platforms, eventually no resources will be left to develop new systems, thereby causing a legacy crisis in organizations (Iyer, 2008). IT budget imbalance between maintenance and innovation is skewed heavily towards maintenance in environments dominated by large legacy systems. This imbalance is often cited as one of the main limitations in freeing-up resources to take a fresh approach to the question of applications agility, which is the capability to rapidly and efficiently adapt applications to change (Tsang, 2002, Gordon, 2004, Heydebreck et al., 2000).

A legacy system is an information system that is built in the past using technology of that time, and that continues to be used, even though the technology has been succeeded by newer technology (Linthicum, 1999). We will use the term existing IT to denote information systems (IS) and information technology (IT) systems, present in the organization's IS/IT landscape, and new IT for IS and IT systems, that are to be procured, sourced or developed in-house. Changes to one part of these existing systems inevitably involve changes to other components. These systems are often maintained because it is too risky to replace them (Bisbal, Lawless, Wu, and Grimson, 1999) and new software development projects are often expensive and risky (Yeo, 2002).

Maintenance and decommissioning of existing systems become a huge problem where business requirements put pressure on the evolution of organization's IS/IT landscape. To provide insight into this issue, we will analyze propositions that influence decommissioning and maintenance of existing systems in this paper.

The salient contributions of this paper are as follows.

- Propositions that influence decommissioning and maintenance of existing IT systems in organizations are identified based on literature study and interviews conducted.
- A methodology, by which the balance between existing and new IT can be measured, describing the governance of existing and new IT systems in organizations, is presented. This methodology contributes to the theory on portfolio management, see for instance (Blume, 1970).

The remainder of this paper is organized as follows. In section 2, we provide the research methodology. In section 3 we describe the findings from literature that influence the decision to decommission or maintain, and we illustrate these propositions with statements from interviews. Section 4 addresses the way to measure the imbalance between existing systems and to-be new systems and proposes a methodology to govern these systems, followed by concluding remarks in section 5.

2 Research Methodology

Propositions influencing maintenance were identified using literature research. As the fields of software maintenance research and organizational economics research exist for decades and a lot of research is still going on, we had a considerable amount of literature for our research questions. We have extensively searched using multiple relevant keywords in various internet resources including IEEE, ACM, Elsevier, and Springer repositories. New articles are also discovered from the bibliographies of relevant publications, and by searching who references a particular relevant publication. Then, we have shortlisted the literature that we considered useful to form our propositions. As literature often mentions these propositions but did not explain them in detail we opted from organizing discussions with five domain experts and conducting twenty face-to-face interviews with high level managers of large IT user organizations. Thus we have applied triangulation (Denzin, 2006) by using literature and interviews with experts and practitioners. The domain experts were selected based on their experiences in the field of maintenance and decommissioning of legacy systems and technology management. Their inputs and suggestions helped us to further refine the propositions. The organizations interviewed include the following. three government organizations, one energy company, one consultancy firm, one pension fund company, two banks, and two insurance companies. These organizations typically have a large IT center to serve the business needs. In total twenty interviewees were interviewed representing ten different organizations.

The interviews were semi-structured using an interview protocol. Each interview took between one to two hours. In the interview protocol the following elements were addressed: introduction, current practices related to costs and benefits of IT, decision-making on decommissioning, the role of architecture, budget spending, the requirements for managing the IT portfolio, and improvement mechanisms for having a healthy balance between existing systems and to-be proposed systems. The interview questionnaire is given in the appendix. Transcripts were made of all the interviews. The interview reports were validated with the interviewees by sending the report to the interviewees and asking them for comments. Then the reports were updated and archived.

Overall we followed the deductive approach by defining propositions from the literature and we followed an inductive approach by verifying these propositions using the semi-structured interviews.

3 Influencing Propositions on Maintenance and Decommission of Existing Systems: Theory and Practice

In this section, we present propositions (P1 – P7), influencing the decommission and maintenance of legacy systems as found in literature. By conducting interviews, we were able to relate these propositions to the practice in large IT user organizations. We used interviews to refine the propositions and selected statements from the interview reports as illustrations for some of the propositions found in literature.

P1— Software characteristics affect system maintenance.

There has been a great deal of speculation about what makes a software system more difficult to maintain. System size, system age, number of input/output data items, application type, programming language, and degree of structure were the characteristics that make system maintenance difficult (Martin and McClure, 1983).

Larger systems require more maintenance effort than smaller systems, because of a lengthier learning curve associated with larger systems, and larger systems are more complex in terms of the variety of functions they performed. Furthermore errors and code, that required changing, are more difficult to find in larger systems.

The length of the source code is the main determinant of total cost during maintenance as well as initial development. For example, a 10% change in a module of 200 lines of code is often more expensive than a 20% change in a module of 100 lines of code (Van Vliet, 2000). There were an estimated 120 billion lines of source code in legacy systems being maintained in 1990 (Ulrich, 1990). According to (Sommerville, 2000), this amount of code doubled (250 billion lines) in 2000.

In general, programs written in high-level languages are easier to understand and to modify than their lower level counterparts. Some programming languages, such as Java, are independent of the operating system, making it easier to adapt the program to changes in its hardware and software environment.

Applying software engineering design principles, such as information hiding and encapsulation, make software easier to maintain. Adopting loose coupling, high cohesion, and fewer internal dependencies facilitate constant change and revision even during development (Parnas, 2011). An illustration for proposition P1: *"Built with data-driven programming languages, generally it is not possible or difficult to extract business content from existing legacy systems"*. The architecture of such existing systems prohibits the design of new solutions to monitor business processes.

P2 — Different system development methods imply different maintenance efforts.

Maintenance is heavily impacted by the methods used to develop a system (Erdil et al., 2003). Iterative development requires maintenance at every cycle when a working system is being developed. This avoids bugs in the systems. Agile development methodologies tend not to favor extensive documentation, thus

complicating maintenance at a later stage. However, agile methodologies signify corrective and perfective maintenance (Cohen, Lindvall and Costa, 2003).

In component-based software development, system developers do not have access to the source code. Hence, maintenance and evolution of the component are controlled by a third party. In a case study (Pree, 1997) it was found that a few changes (12%) on a single component increased the reuse cost by 55% compared to the development of the particular component built from scratch. Reuse may have saved time, but may have also increased the risk of corruption or unreliability due to interactions, making the maintenance process more difficult. A catalogue of the different components had to be kept, and the developers and maintainers needed a good understanding of the different interfaces and the intricacies of the system.

An illustration for proposition P2: "*With their interfaces written with rigid functionality, legacy systems cause business silos",* suggesting that organizations reap higher returns when they reuse existing software components and data. Without adequate interface linking systems, reusability is inhibited. The process behind the existing legacy systems could be exposed as well defined and wrapped as services. Existing legacy systems could be redesigned using services based on service oriented computing concepts. Integration techniques including presentation integration, business logic integration, and data integration extend the life of legacy systems.

P3 — The risk of decommission increases when the dependency of systems increases.

Dependencies between system modules play a major role in maintenance. This is due to the fact that changes propagate through dependencies. When a system module is modified, it is possible that other modules dependent on this module will also need to be modified (Ohlsson et al., 1999). The likelihood of a change affecting other modules increases with the number of dependencies a module has. As systems grow more complex and large, the significance of dependency management grows as well (Guo, 2002).

Impact analysis involves the tracing through any relationships defined on an artifact and identifies the targets of the relationships. This impact analysis results in a list of dependencies that the given artifact depends on. From the perspective of requirements engineering, dependencies on goal, service, condition, time, task, infrastructure, and use are identified (Khan, Greenwood, Gracia and Rashid, 2008). From the business perspective, portfolio dependency, contractual dependency, and change, risk and compliance dependency are crucial for system maintenance (Wegener, 2007). It is very important to explore the interrelationships among existing systems and to analyze the impact of the systems that are to be decommissioned on other systems. One comment of an interviewee illustrates proposition P3: *"Legacy systems are added with new features as required by an organization. New modules are designed to be interconnected with other systems of the organization"*. This statement explicitly shows that legacy systems become difficult to maintain and decommission due to their dependencies with other systems.

P4 — Lock-ins increase the risk of decommission.

Various forms of lock-in by vendors, based on knowledge, contract, and technology, may have made organizational systems dependent on a particular set of vendors or proprietary technologies. Traditional software and component contracts or licenses deny the rights to access future versions by consumers. Most organizations feel that running unsupportable software and hardware systems cost more to maintain than to replace (Cohen, Lindvall and Costa, 2003).

Many organizations prefer to adopt single vendor software and infrastructure (Kauffman and Tsai, 2009). Then their decommissioning becomes heavily dependent on the vendor's product release strategy. Although this causes problems during decommissioning, organizations do not prefer to adopt multiple vendor-based solutions as this increases the number of contracts and maintenance costs.

The effect of lock-ins on the balance between existing and new IT may go either way. Organizations may be forced by vendors to decommission systems because product releases are not supported any more, or the organizations may opt for expensive maintenance contracts for products, which are no longer officially supported by the vendor. An illustration for proposition P4: *"Suppliers enforce renewal and phase out."* This interview statement indicates that once an organization is in a lock-in situation, the organization may be forced to decommission due to the product release strategy of the suppliers. Forced decommission may increase the risk of decommission, because the organization might not be ready to decommission due to other priorities and limited resources.

P5 — The more complex the system, the higher the risk of decommission.

As of 2007 the software industry uses some 600 different programming languages and about 120 kinds of software applications (Jones, 2007). Software applications are built using 26 different development methods that include at least 35 different activities. The software industry faces 24 kinds of complexity and performed 23 different kinds of maintenance activities. The plethora of choices and alternatives in software systems enable one to select whatever may fit the situation at hand. However, these approaches make different silos of systems in an organization and make maintenance complex at later stages.

An empirical survey (Banker, Datar, Kemerer and Zweig, 1993) reports that the high level of software complexity accounts for 25% maintenance costs or more than 17% of total life cycle costs. A study of twenty projects by Reifer, Basili, Boehm and Clark (2003) reports that maintenance costs and complexity increase exponentially as the number of packages increased in a components-based system.

Most organizations already have a complex IT environment. As the costs and risks associated with decommisioning these systems are usually high (Aversano, Canfora, Cimitile and De Lucia, 2001), organizations opt for adding new systems instead.

An illustration for proposition P5 is the following interview statement: *"Logically projects with a high business value and a low risk get a higher priority.*

*"*This statement indicated that some organizations tended to avoid risks, and would only accept risk when the business value was high. For many decommissioning projects, however, the business value was not well known by business managers. When risks were high and business values unclear, decommissioning was avoided.

P6 — The rapid emergence of new technologies makes systems obsolete early.

As new technologies emerge very fast, systems turn obsolete early. Because it is unclear how long new technologies will remain, it is hard to claim that adopting new technologies lowers maintenance costs in the future. Castro-Leon, He, Chang and Peiravi (2009) suggest that costs can be managed through aggressive "treadmill" of technology adoption, but this did not fix the general uptrend. Not many organizations are willing or even capable of sustaining this technology innovation schedule. Those organizations, which could not maintain a high speed of decommissioning, ended up with a higher balance between existing and new IT.

A large number of comments were made showing the support for P6 and its importance. *"When something new is introduced you would need to ask the business the question: What will you replace and what is the added value to the organization?"*. Another interviewee commented *"Decisions on rationalizing the portfolio are not just for the IT department, but must be taken in a broader context. Because no extra functionality is delivered, and because of the focus on going concern, the business is not always convinced of the usefulness of rationalizing."* and *"It proves to be very difficult for the business to say goodbye to systems, since decommissioning also costs money."*

All three statements indicate that the interaction between business and IT is often not sufficient for effective decommissioning. Both business and IT people were often more focused on new functionality and technology rather than on decommissioning existing functionality. The fact that decommissioning of IT systems first requires investments before it saves costs, is sometimes sufficient to prevent decommissioning of systems. One interviewee provided a suggestion for dealing with this.*"In our organization we work with the principle: When something new comes, something else has to go. Earlier we used a ratio that for each new application three existing applications had to be decommissioned, unless the business could convince us to act otherwise."* Using this principle this organization managed to cut down the application portfolio from 3000 applications to 600 over a period of 4½ year. This organization managed the balance between existing and new IT successfully without application or project portfolio management, just by using some simple rules that the business could understand, and by moving along with existing drivers for change in the organization.

P7 — Ownership impacts decommission.

An organization builds systems in-house if these involve confidential business logic and data processing operations that are considered to be of core value to an organization, or when systems on the market do not address the requirements

(Cullen, Seddon and Willcocks, 2005). With some kinds of sourcing patterns and contracts, an organization can completely own a system even if it is not developed in-house (Cullen and Willcocks, 2003). Although the organization has ownership over these systems, one of the costly problems is supporting the legacy software and applications. If the system was developed in-house, the organization should have supporting staff for the system. If the supporting staff was not available, at least the organization should have enough documentation to support it, now or in the future (Marwaha, Patil and Tinaihar, 2006).

Experiences in a large organization with many legacy systems suggest that systems, owned by an organization, may remain operational longer than originally planned. In some cases ownership of a system hampers its change. Sometimes owners tend to prefer maintenance over change.

"The business decides on decommissioning of IS/IT, because the systems are theirs. Cost and business value are estimated together with the business, and the business tells the story in the IT governance board. But in practice very little is decommissioned, amongst other things because that requires investments. As a consequence license costs are rising." This quotation shows that ownership plays a crucial role in decision making on decommissioning of systems. When the owner understands the consequences of late decommissioning, like expensive maintenance and support contracts, higher license costs and functional problems for users, the owner is likely to be more active in decommissioning of IS/IT systems, provided that budget was available to invest in decommissioning. Despite the importance, none of the interviewees stated that their organization was actively governing the balance between budget spend on controlling and maintaining existing IT and money spend on acquiring new IT. Most organizations were taking measures such as portfolio management and creating transparency in budget spending and the related decisions. Portfolio management was often used to determine which projects should be done and which projects should not be done. It did not include mastering the balance between existing and new IT.

4 Methodology for Governing Existing and New Systems

Once an imbalance between existing and new IT is present, organizations may want to improve the balance to ensure that sufficient budget will be assigned to new development and innovation. The normative question remains: What is a healthy balance? There seems to be little evidence on what constitutes a healthy balance in attention and resources between introducing new and maintaining existing systems based on our research described earlier. The healthiness may depend on the nature of the organization at hand, its state of development, maturity of the technology at hand, and other factors.

We propose a methodology by which the balance between existing and new IT can be measured. This methodology can be integrated with existing portfolio management methods. The idea is to make visible what the ratio between existing

and new systems is to create awareness and to let the stakeholders determine what the ratio should be according to their insights. Based on the proposition discussed in the previous section directions for improvement can be determined. As existing systems become legacy this should be a continuous activity and the process is started again. The following sections describe:

1. *Defining and measuring* the balance using an innovation indicator to describe the organization's IT landscape and innovation capability;
2. *Shifting* the balance by using measures related to the factors;
3. *Keeping* the balance by deploying adequate IT management and governance processes.

Defining and Measuring the Balance

Once an organization has analyzed its mission, identified by its stakeholders, and defined its goals, it needs a way to measure progress toward those goals (Reh, 2012). In this paper, we define an *innovation indicator* which is the ratio of budget percentage for existing legacy systems and budget percentage for new IT systems, to govern the balance between existing and new IT in an organization at the strategic level. So in (Glass, 2006) an illustration of this indicator is that if 80 % existing IT and 20 % is new IT then this indicator is 80/20 = 4.

The higher the value of the innovation indicator, the less space there is for innovation of IT (new IT). The ratio between existing and new IT can be used as an indicator for making decisions on decommissioning and maintenance of existing and new IT and for governance purposes of innovation.

Choosing a fraction as the indicator makes clear how serious the situation becomes when the balance is skewed heavily towards existing IT. Then the value of the innovation indicator moves to infinity, suggesting that innovation is squeezed out entirely. In such a situation the organization may not be able to maintain existing systems due to expensive maintenance contracts for out-of-date IT. Organizations need to establish what they consider to be a healthy value for this innovation indicator. For instance an innovative organization may strive after a value of 1, whereas a trend-following organization may strive after a value of 3 for the innovation indicator.

Shifting the Balance

Even if the role of IT in an organization or its operating model provide some reason for relatively high maintenance costs, any organization would favor to lower its maintenance costs, as the released budget can be subsequently used for IT innovation. The propositions presented in section 3 provided the basis for defining strategies and measures, which favor a healthy balance between existing and new IT. In Table 2, an overview of strategies and measures to improve the balance between existing and new IT are shown which are based on industry practices obtained from the interviews and literature review.

Table 2. Measures to improve the balance between existing and new IT related to propositions.

Propositions	Improvements
P1. Software characteristics	• Prevent systems from becoming too large and complex. • Use high level languages. • Apply software engineering principles.
P2. Development methods	• Select and use development methods with a view on the impact on maintenance and decommissioning of systems.
P3. Dependency of systems	• Aim at loose coupling between systems and services. • If a system has more dependencies, wrapping may provide a solution.
P4. Lock-ins	• Use open standards where available.
P5. System complexity	• Restrict the size of systems. • Keep a minimum level of granularity of control over the IS/IT landscape.
P6. New technologies	• Assess the stability of new technologies and delay introduction when needed.
P7. Ownership of systems	• Make business owners aware of the costs and risks of late decommissioning.

Keeping the Balance

Organizations need to monitor the balance and take appropriate measures if needed. In our interviews three type of measures were identified that can add to an organization's capability of balancing existing and new IT in its IT budget:

- *Making budget allocation decisions persistent and consistent across the entire life cycle.* Total Cost of Ownership (David, Schuff and St. Louis, 2002) is a well-known approach using this principle. But also servitization (Turner, Budgen and Brereton, 2003) may bring along this persistence, as services tend to involve explicit service contracts, with life-time costs included in the contract prize. Setting such prices requires the service providers to look ahead and think about maintenance and decommissioning beforehand.
- *Explicit IT portfolio management.* This makes transparency between the project portfolio and the way budget is spent. Portfolio management may use the distinction between maintenance and innovation projects. Additionally, it may impose policies, restrictions and roles to the balance. Such rules should be grounded in a solid understanding of the role of IT in the organization and the organization's operating model.
- *Enterprise architecture management.* The management of enterprise architecture may help rationalizing IT budget decisions (Ross, Weill and Robertson, 2006).

Especially where extensive interdependencies across many systems and services deprive an organization of the courage to decide to change or decommission systems and services, enterprise architecture provides clarity and prevents a maintenance cost-raising spiral.

Once an imbalance between existing and new IT is present, organizations may want to improve to allow for sufficient innovation. There seems to be little evidence on what constitutes a healthy balance in attention and resources between introducing new and maintaining existing systems. The healthiness may depend on the nature of the organization at hand, its state of development, maturity of the technology at hand, and other factors. A healthy balance could be determined at the strategic level of the organization. Once the innovation indicator is used, benchmarking with other organizations can help further to establish the desired value.

5 Concluding Remarks

Theory and practice show that there is a gradual increase in maintenance costs over the years, rising to 90% of the budget in some exceptional cases. The inability to manage the IT portfolio results in the innovation squeeze, in which increasing portions of the budget are spent on existing legacy systems and thus cannot be spend on innovation. Some large IT user organizations with many legacy systems are faced with the problem of an imbalance between existing and the need for adopting new IT. Their CIOs find themselves in a position where they require more budget every year for IT because projects take longer and cost more than originally anticipated, and the existing IT landscape is consuming an ever increasing percentage of the budget.

Using literature and interviews seven propositions were derived suggesting that software characteristics, development methods, dependency of systems, lock-in, system complexity, new technologies and system ownership influence the decision whether to decommission or to maintain a system. The balance between existing and new IT in an organization can be improved by making use of the seven propositions to determine the influences and then select the most effective measures.

Our interview results indicated that the balance between existing and new IT was not effectively mastered in most of the organizations interviewed. We developed a methodology that recommends CIOs and CFOs to start a discussion where they establish what they find a healthy balance between existing and new IT for their organization based on an innovation indicator. They can use this innovation squeeze indicator to find out what the current value is, and to discuss what the future value should be. This should help to initiate a process to analyze what the main influences are on the balance between existing and new IT in their organization and to ensure that the balance between control, maintenance and innovation is maintained. We did

not use this methodology in practice and recommend to evaluate its value in further research.

More research is necessary on decommission strategies to prevent that organizations become trapped by the maintenance costs of existing IT and can hardly innovate. More research is necessary to test the methodology and the use of the innovation indicator to govern the balance between new and existing systems described in this paper.

Acknowledgement

This work resulted from the ArchiValue project, in which the Dutch Tax and Customs Administration collaborates with Novay, BiZZdesign, and APG.

REFERENCES

Aversano, L., Canfora, G., Cimitile, A. and De Lucia, A. *Migrating Legacy Systems to the Web: An Experience Report.* In: Proceedings of the Fifth European Conference on Software Maintenance and Reengineering, Lisbon, Portugal, March 14-16, 2001, pp. 148-157.

Banker, R., Datar, S., Kemerer, C. and Zweig, D., 1993. Software Complexity and Maintenance Costs. *Communications of the ACM,* 36 (11), pp. 81-94.

Bisbal, J., Lawless, D., Wu, B., and Grimson, J., (1999). Legacy Information Systems: Issues and Directions. *IEEE Software*, 16 (5), pp.103-111.

Blume, M.E., 1970. Portfolio Theory: A Step Towards Its Practical Application. *The Journal of Business,* Vol. 43, No. 2, pp 152-173.

Castro-Leon, E., He, J., Chang, M. and Peiravi, P., 2009. *The Economics of Service Orientation.* Available from: <http://www.infoq.com/articles/intel-services-economics>. [24 November 2011].

Cohen, D., Lindvall, M. and Costa, P., 2003. *Agile Software Development.* New York: Data and Analysis Center for Software (DACS).

Cullen, S., Seddon, P. and Willcocks, L., 2005. Managing Outsourcing: The Lifecycle Imperative. *MIS Quarterly Executive,* 4(1), pp.225-246.

Cullen, S. and Willcocks, L., 2003. *Intelligent IT Outsourcing.* Oxford: Butterworth Heinemann.

David, J.S., Schuff, D. and St. Louis, R., 2002. Managing your total IT cost of ownership, *Communications of the ACM*, 45 (1), pp. 101-106.

Denzin, N., 2006. *Sociological Methods: A Sourcebook.* Aldine Transaction.

Erdil, K., Finn, E., Keating, K., Meattle, J., Park, S. and Yoon, D., 2003. Software Maintenance as Part of the Software Life Cycle. Technical report, Department of Computer Science, UFTS University.

Erikh, L., 2000.Leveraging legacy system dollars for e-business. *IEEE IT Professional*, May/June, pp. 17-23.

Eastwood, A., 1993. Firm fires shots at legacy systems. *Computing Canada,* 19 (2), p. 17.

Gangadharan, G.R., Kuiper, N. and Oude Luttighuis, P., 2010. *Balancing Old and New in IT Budgets.* Novay Technical Report. Available from: <https://doc.novay.nl/dsweb/Get/Document-114641>. [October 2012].

Glass, R.L., 2006. Software Conflict 2.0: *The Art and Science of Software Engineering*. Atlanta: Developer Books.

Gordon, R. J., 2004. Five Puzzles in the Behavior of Productivity, Investment, and Innovation. NBER Working Paper No. 10660.

Guo, J., 2002. *Using Category Theory to Model Software Component Dependencies*. In: Proceedings of the Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, Sweden. pp. 185-192.

Hamilton, D. (1999). Linking stategic information systems concepts to practice: System integration at the portfolio level. Journal of Information Technology, Vol. 14, pp. 69-82.

Heydebreck, P., Klofsten, M., and Maier, J., 2000. Innovation support for new technology-based firms: the Swedish Teknopol approach. Journal of R&D Management, 30 (1), pages 89–100. Hillenius, G., 2006. Taking the measure of IT. *The Economist.* March, 2006.

Iyer, V.N., 2008. Legacy Modernization. Technical Report. Infosys, India.

Jeffery, M. and Leliveld, I. (2004). Best Practices in IT Portfolio Management. MIT SLoan Management Review, vol. 45, No. 3, pp. 41-49.

Jones, C., 2007. *Estimating Software Costs.* New York: McGraw Hill.

Kantner, L., Shrover, R. and Rosenbaum, S., 2002. *Structured Heuristic Evaluation of Online Documentation.* In: Proceedings of the IEEE International Professional Communication Conference, Portland, Oregon, Sept. 17-20, 2002, pp. 331-342.

Kauffman, R.J. and Tsai, J.Y., 2009. *When is it Beneficial for a Firm to Pursue a Unified Procurement Strategy for Enterprise Software Solutions?* In: Proceedings of the 42nd HICSS, 5-8 Jan. 2009, Big Island, Hawaii, pp. 1-10.

Lientz, B. and Swanson, E.B., 1980. *Software Maintenance Management.* Boston: Addison-Wesley.

Linthicum, D., 1999. *Enterprise Application Integration*. Addison-Wesley, MA.

Lowe, S., 2009. *Software Maintenance Pricing – Fair or Out of Control?* Available from: <http://blogs.techrepublic.com/tech-manager/?p=956>. [24 November 2011].

Martin, J. and McClure, C., 1983. *Software Maintenance: The Problem and Its Solutions.* Englewood Cliffs: Prentice Hall.

Marwaha, S., Patil, S. and Tinaikar, R., 2006. *The Next Generation of In-house Software Development*. McKinsey Report.

McFarlan, F.W. (1981). Portfolio Approach to Information Systems, Harvard Business Review, Vol 59, pp.142-150.

Ohlsson, M. C., von Mayrhauser, A., McGuire, B., and Wohlin, C., 1999. *Code Decay Analysis of Legacy Software through Successive Releases*. In: Proceedings of the IEEE Aerospace Conference, USA. 69-81.

Parnas, D., 2011. Software Engineering - Missing in Action: A Personal Perspective. *IEEE Computer*, Vol. 44, No. 10, pp. 54-58.

Port, O., 1988.The software trap – automate or else, *Business Week,* 3051 (9), pp. 142-154.

Pree, W., 1997. Component-Based Software Development – A New Paradigm in Software Engineering? *Software Concepts and Tools,*18 (4), pp. 169-174

Reh, F.J., Key Performance Indicators – *How an organization defines and measures progress towards its goals*. Available from: <http://management.about.com/cs/generalmanagement/a/keyperfindic.htm>. [20 March 2012].

Reifer, D.J., Basili, V.R., Boehm, B.W. and Clark, B., 2003. Eight Lessons Learned during COTS-Based Systems Maintenance. *IEEE Software,*20 (5), pp. 94-96.

Ross, J., Weill, P. and Robertson, D., 2006. *Enterprise Architecture as Strategy: Creating Foundation for Business Execution.* Boston: Harvard Business School Press.

Sommerville, I., 2000. *Software Engineering*. Boston: Addison-Wesley.

Tsang, A.H.C., 2002. Strategic dimensions of maintenance management", Journal of Quality in Maintenance Engineering, 8 (1). pp.7 - 39.

Turner, M., Budgen, D. and Brereton, P., 2003. Turning Software into a Service. *IEEE Computer*, October 2003, pp. 38-44.

Ulrich, W., 1990. The Evolutionary Growth of Software Engineering and the Decade Ahead. *American Programmer,* 3 (10), pp. 12-20.

Van Vliet, H., 2000. *Software Engineering: Principles and Practices*, 2nd Edition. West Sussex: John Wiley and Sons.

Yeo, K. T., 2002. Critical failure factors in information systems projects. *International Journal of Project Management*, 20, 241-246.

Appendix: Interview Questionnaire

1. Do you have a record of budget details for existing IT projects and projects that are planned in the near future (new IT) in your organization?
2. Do you discuss the expenses for existing IT and new IT projects?
3. How are costs allocated? How are benefits attributed?

4. In what terms is the value of IT defined and quantified?
5. How do you quantify the value/benefits of IT systems versus expenses?
6. Are there also other ways (e.g. Balanced Score Card, Information Economics, Multi criteria, economic indicators such as TCO, ROI, etc. ...) used? Why are these ways (no longer) used?
7. What are your experiences with the present approach? Explain with examples.
8. In what manner do the cost benefit calculations for investment decisions change new IT proposals? Are there any risks associated with cost / benefit weighted?
9. How do you estimate cost benefit calculations for managing IT project portfolio?
10. How is it decided to decommission or replace IT?
11. Are decisions, prioritization rings and business cases evaluated at a later time?
12. Could you please indicate what percentage (approximately) of the budget will be devoted to experiments - development of new IT - maintenance -phasing?
13. Could you please indicate which percentages (approximately) of the budget are spent on experiments - development of new IT – maintenance - decommissioning?
14. How will you focus on the weaknesses of existing IT: to remedy, or is it to future business opportunities?
15. How is your business value driven: by physical, or intellectual resources?
16. Is your business model influenced by new technology or not?
17. Do you see whether IT outsourcing is a threat to your operations, or is outsourcing establishing a balance between IT insourcing and outsourcing?
18. What are the criteria that you will use for valuing IT? Do you use a rigid criterion or multiple criteria to estimate the value of IT?
19. In what way would you like to manage your IT portfolio?
20. How would you decide to invest in experiments, IT development, maintenance, and decommissioning?