Effective Monitoring and Control of Outsourced Software Development Projects

María Laura Ponisio[1] & Peter Vruggink[2]

[1] University of Twente The Netherlands ml@ponisio.com

[2] Logica The Netherlands peter.vruggink@logica.com

Abstract: In our study of four outsourcing projects we discover mechanisms to support managerial decision making during software development processes. We report on Customer Office, a framework used in practice that facilitates reasoning about projects by highlighting information paths and making co-ordination issues explicit. The results suggest a key role of modularisation and standardisation to assist in value creation, by facilitating information flow and keeping the overview of the project. The practical implications of our findings are guidelines for managing outsourcing projects such as to have a modularised view of the project based on knowledge domains and to standardise co-ordination operations.

Keywords: Software development, outsourcing, project management

1. Introduction

Software development involves both managerial and technical decisions. Extensive research has been done on concepts and best practices to improve project management, such as software cost estimation models (*e.g.,* COCOMO), development process frameworks (*e.g.,* RUP), maturity models (*e.g.,* CMMI), governance models (*e.g.,* Cobit), and project management methods (*e.g.,* Prince2).

However, in practice, project managers drive projects based mostly on experience. They still find it difficult to know which mechanisms are useful to control large software development projects. Specifically, practitioners ask for effective mechanisms to control software projects and to connect strategic, technical and organisational domains.

Outsourcing software development presents extra challenges because development is performed in an inter-organisational context. A customer (an organisation) asks a vendor (another organisation) to produce some IT artefact, such as a software application. Although organisations collaborate transferring knowledge from customer to vendor (*e.g.,* requirements) and from vendor to customer (*e.g.,* appropriate technology), each has its own interests and needs; which often conflict. Tacit requirements, conflicting interests and knowledge-domain gaps add up to

18th International Conference on Information Systems Development (ISD2009)
Nanchang, China. September 16-19, 2009

generate final solutions that cost more in terms of resources than what was originally planned or that do not help customers to meet their ambitions.

The objective of this work is to understand mechanisms that organisations put in place to optimise software development management in outsourcing projects. In particular, this work aims to gain knowledge about which mechanisms facilitate transfer of the information that managers need when making decisions during software development.

We have studied four projects representative of software development in an outsourcing context. The results suggest a key role of modularisation and standardisation as effective mechanisms to manage development projects towards value creation.

Customer Office (CO), a reasoning framework found in practice in the observed organisation, serves as an example to explain this modularity observed in practice. CO has the potential to be an effective operationalisation of CMMI for acquisition module because it highlights information paths and makes co-ordination issues explicit. Prioritising reasoning about the project in terms of working units, CO encourages project managers to keep the oversight of co-ordination. Putting co-ordination up-front, project managers have better control of the information flow, which is vital to transfer requirements effectively and to gather technical information necessary to make informed decisions.

The practical implications of our findings are guidelines that include: have a modularised view of the project according to knowledge domain and standardise co-ordination operations. At the organisational level, results point at the importance of having managers with *experience* and of providing managers and engineers with managerial information that managers can relate to.

2. Maximising value creation in software development

To maximise value creation project managers must understand the connections between technical decisions and enterprise-level value. With inadequately understood connections, project managers are unable to make decisions that could significantly increase the value created by software development. Consider software modularity. The ability to meet time-to-market requirements depends on having a modular design. An independent-feature-based architectural style helps developers to meet time-to-market requirements because it enables them to abandon unimportant features later if times runs out. Project managers in software development strive, thus, to connect technical decisions with value creation criteria. Practitioners ask for mechanisms that help them to connect strategic, technical and organisational domains. In other words, mechanisms that help them to answer the question **how are we in control?**

2.1. Related work and existing solutions

Solving the technical-value mismatch occurs in the context of two domains: software development (in the area of software engineering) and management in IT outsourcing projects (in the area of IT project management). Decision making is the linking pin between these domains because better decision making is a key enabler of business-IT alignment. The following sections elaborate on the related work in these areas.

2.1.1. Support for decision making in software development

One way to exercise control is to connect technical properties of the software product with decisions supporting value maximisation. Managers do this by measuring properties of the software (to follow closely the development needs), by estimating the cost (or effort) required to develop a system, and by choosing good models for software development. The next paragraphs elaborate on these issues.

Software metrics. This term describes a range of activities related to measurement in software engineering. These activities include measuring some property of the software code (such as size and complexity) and quantitative aspects of quality control and assurance [5].

One of the major reasons for using software metrics is to improve the way decision makers monitor, control and predict various attributes of software and the software development process. Furthermore, metrics are also used to measure software product quality and evidence of use of metrics is needed to achieve higher levels of the Capability Maturity Model (CMM).

The major problems of metrics are using measurements in isolation, handling uncertainty, and combining them with evidence. After 40 years of research, we have learnt that as important as the metric is its *application*. Metrics must, thus, have clear goals and objectives.

Software cost estimation models. Software cost estimation is concerned with making estimates of the effort required to complete the software for a system development project [15]. Research in the area of software engineering and economics has produced a number of software cost estimation models such as COCOMO, PRICE S, SLIM, SEER, SPQR/Checkpoint, and Estimacs. They are still in use today. A recent systematic review identified 304 software cost estimation papers in 76 journals [7].

In spite of much theoretical support to reason about software cost, support to reasoning about benefits and value is sub-optimal. The models need to adapt to support reasoning about benefits and cost in a context that rapidly shifts priorities. For instance, estimation models need to be integrated to the current inter-organisational way of developing software. Outsourcing introduces *multiple stakeholders* whose conflicting interests need to be addressed. Outsourcing software development by definition involves co-operation among several organisations (see Figure 1). Development in an inter-organisational context challenges the

organisation's operating model – the necessary level of business *process integration* and *standardisation* for delivering goods and services to customers.

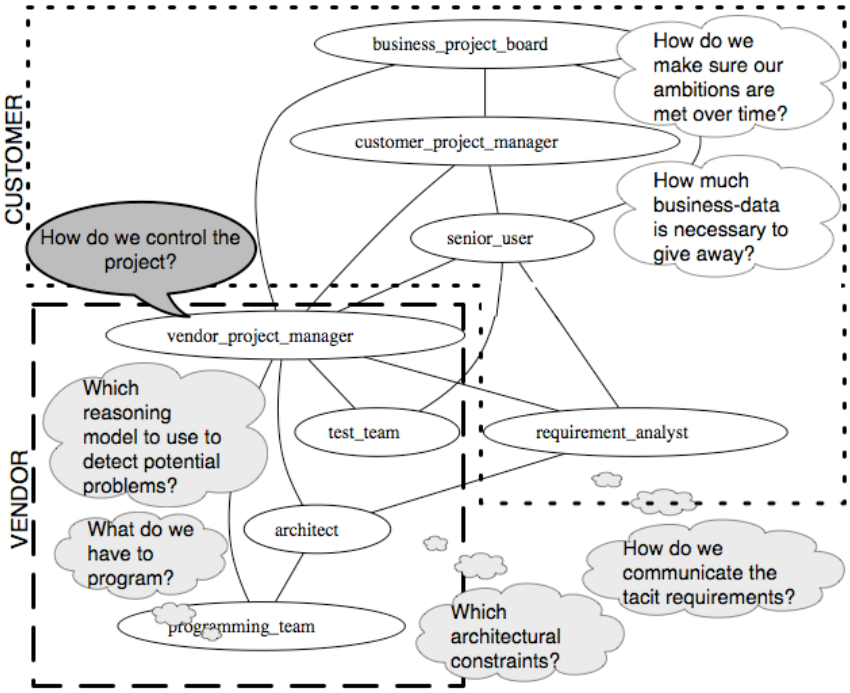

Figure 1: Outsourcing actors forming an inter-organisational network to carry on an IT solution-development project.

Software development models. They correspond to methods used to support software development activities. Several solutions from perspectives ranging from requirement techniques to agile development have been proposed. For example, V-Model, Spiral model, Waterfall model, RUP, Iterative model, Agile, Scrum, and eXtreme programming). All these methods and techniques have advantages and disadvantages. Success in their application depends on the project characteristics and the way they are applied. Unfortunately managers have very few guidelines regarding ways to operationalise them.

All in all, we observe that results from software engineering are not presented in terms that clearly show value for controlling software development. It is understandable that practitioners express the need for more practical mechanisms.

2.1.2. Management in IT outsourcing projects

Scholars try to understand governance from two main different perspectives: connecting business and technology (from a business Information Technology (IT)

alignment perspective) and managing the project (from a governance perspective). In both perspectives outsourcing introduces new socialisation and co-ordination issues related to the customer vendor relationship.

Business IT alignment. Establishing the connection between technology objectives and business objectives impacts on the value of the software development results [10]. The field is very active. A recent literature review on IT alignment [3] describes over 150 alignment articles. Research highlights the importance to share responsibility for alignment, build the right culture, educate and equip, share knowledge, and manage the IT budget. It has been pointed out that needs for objective alignment differ between organisations [13] and available governance options have been described [11].

However, little research investigates how, when, and where we can improve IT alignment. In fact, according to Gutierrez et al. "the variety of approaches proposed has created confusion about the applicability and context in which these approaches can be used" [6]. That confusion explains why, in practice, managers rely mostly on their experience to choose the adequate theoretical constructors and their practical use in a specific project.

Governance in IT outsourcing. According to Weill there are five areas where IT decisions have to be made (IT principles, IT architecture, IT infrastructure strategies, business application needs, and IT investment and prioritization) and that there are several styles. Top-performing firms use particular combinations [16]. Many concepts and best practices have been proposed (*e.g.,* ASL, BiSL, DSDM, RUP, CMMI, CobiT, Prince2, and ITIL). While existing research has produced useful models and propositions, there is need for empirical research that bridges the gap between software development and governance.

All in all, we can observe that many concepts and best practices have been proposed. Yet, a major problem remains: it is unclear which mechanism to use. In spite of significant research efforts, there is need for practical findings that help practitioners to know when, how and where can they exercise control in outsourced software development projects.

3. Research method

We carried out a case study. Drawing from established processes of investigation aimed at the discovery of facts [4], case study research is appropriate to investigate the how and why of projects occurring in reality. Klein and Meyers demonstrate that case study research can be interpretive and indicate seven principles of interpretive research [9]; which we followed in our case study.

According to Klein and Meyers, interpretive research attempts to understand phenomena by gaining knowledge of reality through social constructions, documents, tools, and other artefacts. Interpretive research focuses on the complexity of human sense making as the situation emerges rather than predefining dependent and independent variables. It is appropriate for our research because our problem

consists of gaining knowledge. It helps us, thus, to understand phenomena by interpreting the software development situation as it emerged in practice.

3.1. Research design

We are interested in finding how managers deal with software development issues such as complexity and requirements transfer across domains. First, we ponder how the elements controlling offshore outsourcing software development interact in real-life projects. For instance, we consider which artefacts (such as Requirements Management Plan, Software Architecture Document, Master Test Plan, Use Cases, and Financial Status Report) were mandatory for the delivery of standard offshore, outsourcing development projects, and which artefacts were essential in practice in such projects.

Second, we studied documentation of such projects and a best practices framework used in the organisation under study. In doing so, we focussed on finding actors, artefacts, team organisation, information exchange and dependencies. We paid special attention to recurrent problems (such as tacit requirements) and counter-acting measures. To that regard we consulted experts so we could learn their opinions and tips. We observed four large offshore outsourcing projects and analysed any mechanism we could observe.

Finally, we focused on any mechanism that seemed novel (against the study of the state of the art) and potentially effective to control these projects. Since it was an iterative study, observation and analysis made us gain knowledge of reality, understanding the phenomena. Eventually, all the details acknowledged were in harmony and the moment of understanding had arrived for us. To prevent misunderstandings and thread to internal validity, once we had our findings, we confirmed them by way of interviewing expert project managers. See Section 5.2.

3.2. Data collection method

Data was collected using a combination of methods: interviews, unstructured observations of documentation, and focus groups. According to Eisenhardt [4], collecting data from different types of sources gives more strength to validity. Thus, the evidence was gathered from different sources, where evidence found in one source (*e.g.,* interviews) corroborates evidence found in another (*e.g.,* observations and focus groups).

The observations took place between January 2008 and March 2009 and were made by the two researchers. To double check the findings and to detect potential misunderstandings, focus groups and extra interviews with three experts were performed. The experts were software architects each with more than ten years of experience in managing software development projects. A semi-structured interview protocol was used, to allow the participants to clarify terms and to investigate issues that could improve the description of the situation. Participants were guaranteed anonymity and information was sanitized.

3.3. Case selection

Our case study is based on four projects executed in an organisation that we name Big. Being a multinational IT service provider, Big has offices in many countries. Big is a good example of information technology (IT) service provider because it exemplifies most (large) offshore outsourcing development organisations. In the Netherlands, with several thousands of employees, Big is in the top-ten of IT service providers ranked to number of employees and revenue.

The four projects that we observed in Big reproduce the state and behaviour of traditional outsourcing projects, and it is reasonable to consider these projects representative of their kind. These projects are examples of offshore in-house development of software on behalf of Big's customers carried out by geographically distributed teams formed by 10 to 46 members.

4. Case Study

The projects observed develop information technology (IT) solutions in the context of outsourcing. As such, they evolve around the relation between a customer, a domestic (on-site) team and an offshore team. Both teams belong to Big. The difficulties that managers of Big experience are related to problems to control large complex projects with geographically (and culturally) distributed teams. In particular, challenges are related to large project size and complexity, and requirements transfer (*e.g.*, sub-optimal understanding of tacit requirements or too late understanding).

Since these projects are large, customer and vendor need to put in practice mechanisms to (a) manage the complexity of designing in the large and to (b) maintain the consistency between business goals and the system's architecture in spite of having multiple teams geographically distributed. Big's strategy with these projects is to be customer oriented and to divide-and-conquer. This strategy manages their strategic goal to build long-term partnerships with the customer. In practice, this means to have part of the staff working closely with the customer and to govern the project emphasising the client's view. In the next sections we explain the practices we observed.

4.1. Customer Office

Recently, project managers implemented mechanisms to optimise management of software development in outsourcing projects. Among those mechanisms we find a reasoning framework called Customer Office (CO). This framework was developed bottom-up by the second author of this paper and his team.

CO is a reasoning framework based on Result Delivery (RD), a model inspired by Prince2 [1,2]. RD is organised around key process areas of CMMI acquisition. Figure 2 depicts the RD model. Practitioners implemented the CMMI acquisition module and operationalised it through CO. In the documentation, we observed that

RD encourages organising the working force into *units* (*e.g.*, development centre, business project board, IT project board) organised by *knowledge domain*.

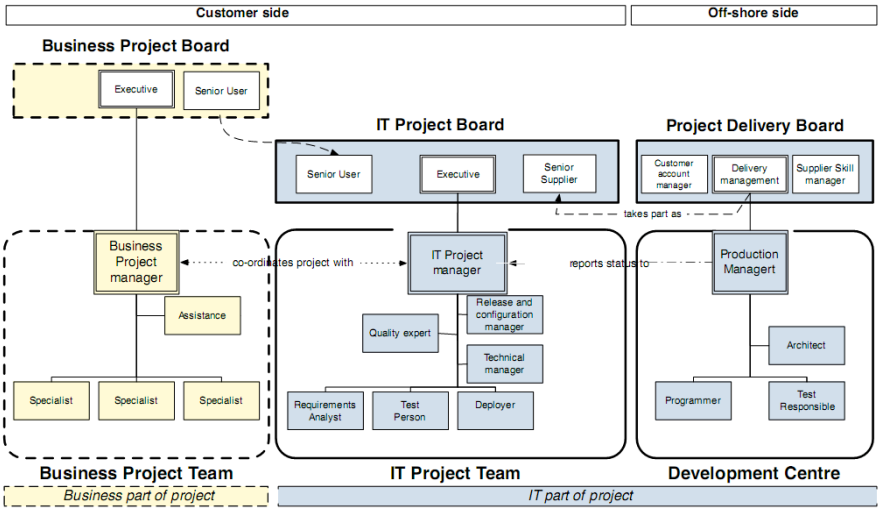

Figure 2: The Result delivery (RD) model.

The CO framework places the IT project team in the centre of co-ordination activities, helping managers to derive IT solutions that meet the customer's business goals by improving the link between the Customer and the Development Centre.

> "CO emphasises collaboration issues with the client (who has the business know-how) and smoothes the way for us to focus on value added to the overall business process. **The client is always responsible for the project.** [Emphasis added by the interviewee.]"

As Figure 2 shows, this organisation minimises coupling between units concerned with different domains (such as managers and developers), maximising cohesion among units (*e.g.*, a development unit was devoted to programming one package). Coupling and cohesion are attributes that describe communication across units and the relationships within a unit, respectively.

4.2. Experience

The main challenges of the observed projects are their size and complexity. Geographical and domain knowledge distance hinder effective requirements transfer across business and technical domains. Moreover, size makes it difficult to recognise essential data and to keep the overview of activities. Recognising, for example, which of the 72 encouraged documents is essential to detect potential problems and opportunities to increase value is performed mostly on the basis of the manager's *experience*.

> "We know that we have to make sure the following reports are made [thoroughly]: Financial Status Report, Quality Status Report, Project Status Report, Customer Status Report and summary of both."

These are the reports managers find most important. One explanation is that managers can relate to these documents because their contents have meaning for them. These reports provide data that managers use to discuss with the customer. In particular, the domain knowledge related to these documents corresponds to the domain knowledge of the IT Project Board unit; which is close to the domain knowledge of the Business Project Board unit.

We observed that in general, project managers have years of experience. Of course, generating experience demands time. In Big employees are encouraged and supported to assist to training that is expected to benefit both their job and long-term career plans.

5. Results

The results of our study show that various mechanisms were used in those projects. Means through which the observed organisation optimises management of software development projects are modularisation of activities and standardisation of co-ordination mechanisms.

Modularisation facilitates control over co-ordination across units because it prevents jeopardising the connections between technical issues and business goals (which hinder value creation if badly managed).

The realisation of this element in the project corresponds to the unit structuring, specifically, as observed in RD and operationalised in CO. Viewing the project in terms of RD's units facilitates *reasoning about the project* because it highlights the information paths and it makes explicit co-ordination issues. In other words, prioritising reasoning about the project in terms of these working units, RD encourages project managers to pay attention to co-ordination. Putting co-ordination up-front, project managers have better control on the information flow, which is vital to the effective transfer of requirements and to make informed decisions. We, thus, interpret CO as an effective mechanism to increase co-ordination among the units; which is essential to deliver software products with value.

In addition, results highlight the importance of supporting experience by providing managers with managerial information that they can relate to, *i.e.,* information that has meaning to them because it is at least partially within their knowledge domain. In our study, we observed that among the 72 suggested documents, only five were key (according to the project managers).

We learn that in Big transferring technical information to managers works well because they have found mechanisms to co-ordinate effectively all the project units: cross-team communication is enhanced by a modularised organisation, favouring co-ordination, and by standardised co-ordination practices, favouring predictability and repeatability in translating across knowledge domains, languages

and cultural factors. The practices observed match common challenges of geographically distributed teams and are in line with their CO reasoning framework.

5.1. Practical implications

Have a modularised view of the project. A modularised organisation happens when there are standardised procedures to communicate information across well-defined units. Actors within a unit share knowledge domains. Links across units are few and explicit. RD makes units and their links explicit.

In our observations modularisation was implemented by structuring activities into the diverse units (*e.g.,* development centre, business project board, IT project board). We interpret that in Big modularisation was effectively used to find ways to maximal value throughout the decision space. Managers make better decisions because they have the information they need: there are explicit roads to share cross-domain information.

One way to implement modularisation of units is to follow the RD model. We interpret that RD responds to the manager's needs to control customer-oriented projects. Specifically, it can be an effective view to reason about projects where customer and vendor are seeking long term partnership. Under these circumstances, the RD view helps managers to maintain consistency between customer's business goals and IT solution because it facilitates communication between different groups of stakeholders. This scenario, in turn has the potential to facilitate quality requirement transfer and to support informed decision-making.

Standardise co-ordination operations. Standardisation of operations happens when an organisation puts in place standards (*e.g.,* mandatory documents with explicit communication protocols) intended to improve predictability and repeatability. The realisation of this element in our case study corresponds to their reasoning framework: CO is organised around key process areas of CMMI acquisition module. For instance, every unit has a responsible person and everybody (also people from other units) knows who has been appointed to this task. Therefore, every team member knows 'whom to call'.

The rationale behind standardisation is that it facilitates communication because the processes in place are made explicit (*e.g.*, who is responsible for what). Co-ordination, speed to market and flexibility to change are also improved. The flexibility obtained is paramount in a context signalled by inter-organisational development.

5.2. Validation

External validity (can it be generalised to other cases) is something we cannot prove, but the results of this study are encouraging at the very least. Our findings are based on the study of only four projects and therefore the criterion of transferability is limited. Nevertheless, we believe the results of our study could be generalised for the following reasons: first, the organisation and the projects observed

are representative of software development outsourcing projects, secondly, because our results are in line with existing research [14,12,8], and thirdly, because the outsourcing experts we interviewed can relate to them.

Internal validity (do our interpretations lead to the right conclusions in this case study?) has been established as high by experts. According to them, our interpretations led to the right conclusions; which means that the internal validity criterion is met. The study approach deserves some comments. Klein and Meyers [9] have suggested a set of principles for the conduct and evaluation of interpretive field research in information systems. The research presented in this paper followed Klein and Meyer's principles.

6. Conclusion

This paper reports on a study of four outsourced software development projects. The objective of the study is to learn the mechanisms that organisations put in place to control software development projects in the context of outsourcing.

The results show that various mechanisms were used in those projects. Means through which the observed organisation optimises management of software development projects are modularisation of activities and standardisation of co-ordination mechanisms. Project managers reason about the project in terms of units organised by knowledge domains. This organisation has the advantage to make co-ordination issues explicit, improving keeping the overview of the project.

A second contribution of this paper is reporting on CO, a reasoning framework found in practice during our observations. CO was developed bottom-up by the second author and his team and supports a modularised view of the project. In it, practitioners implemented and operationalised the CMMI acquisition module. Making co-ordination issues such as information paths explicit, CO empowers project managers with a reasoning tool to better control transfer of requirements, and to gather technical information necessary to make informed decisions.

We believe CO has the potential to facilitate management, supporting modularisation and standardisation in the multiple domains of outsourcing projects; which helps managers to keep the project's overview. Future work will determine if CO effectively supports dynamic monitoring and control of complex software development activities.

ACKNOWLEDGEMENTS Supported by The Netherlands Organisation for Scientific Research, project nr. 638.004.609 (QuadRead).

References

1. (2005a). *Managing successful projects with PRINCE2* (4th edition ed.). The Stationery Office. PRINCE2 Manual.
2. (2005b). *Projects in controlled environments (prince)*. Office of Government Commerce (OGC). Accessed March 13, 2009. http://www.ogc.gov.uk/methods_prince_2.asp

3. Chan, Y. E., & Reich, B. H. (2007a). IT alignment: an annotated bibliography. *Journal of Information Technology*, 22, 316-396.
4. Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review*, 14(4), 523–550.
5. Fenton, N. E., & Neil, M. (2000). Software metrics: roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pp. 357–370. New York, NY, USA: ACM.
6. Gutierrez, A., Orozco, J., & Serrano, A. (2008). Developing a taxonomy for the understanding of business and IT alignment paradigms and tools. In Acton T., Conboy J. & Golden, W. (Eds), *Proceedings of the Sixteenth European Conference on Information Systems*. In-print, National University of Ireland, Galway.
7. Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, 33(1), 33–53.
8. Kernkamp, R. (2007). *Alignment of Requirements & Architectural Design In a Blended Delivery Model*. Master's thesis, University of Twente.
9. Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1), 67–93.
10. Luftman, J. (2000). Assessing business-IT alignment maturity. *Communications of AIS*, 4.
11. Miranda, S. M., & Kavan, C. B. (2005). Moments of governance in IS outsourcing: conceptualizing effects of contracts on value capture and creation. *Journal of Information Technology*, 20(3), 152–169.
12. Ponisio, M. L., & Vruggink, P. (2008). Analysing boundary objects to develop results that support business goals. *2008 International Conferences on Computational Intelligence for Modelling, Control and Automation; Intelligent Agents, Web Technologies and Internet Commerce; and Innovation in Software Engineering,* pp.516-521. Los Alamitos, CA, USA: IEEE Computer Society.
13. Reich, B. H., & Benbasat, I. (1996). Measuring the linkage between business and information technology objectives. *MIS Quarterly*, 20(1), 55–81.
14. Ross, J. W., & Westerman, G. (2004). Preparing for utility computing: The role of it architecture and relationship management. *IBM Systems Journal*, 43(1), 5–19.
15. Walkerden, F., & Jeffery, D. R. (1997). Software cost estimation: A review of models, process, and practice. *Advances in Computers*, 44, 59–125.
16. Weill, P. (2004). Don't just lead, govern: How top-performing firms govern IT. *MIS Quarterly Executive*, 3(1), 1–17.