Metrics-based control in outsourced software development projects

Laura Ponisio[1] *and Pascal van Eck*[2]

[1] **BE Software Design, Okeghemstraat 6, 1075 PM Amsterdam, The Netherlands**
Email: ml@ponisio.com
[2] **Department of Computer Science, University of Twente, P.O. Box 217**
7500 AE Enschede, The Netherlands. Email: pascal@pascalvaneck.com

Abstract

Measurements have been recognized as vital instruments to improve control in outsourced software development projects. However, project managers are still struggling with the design and implementation of effective measurement programs. One reason for this is that although there is a large body of research literature on metrics, practical guidelines for choosing among concrete measurements are scarce. We address this gap between research and practice by synthesising knowledge from frameworks and guidelines presented in the software process improvement literature. Our contribution comprises a framework that provides a set of measurements (selected from the research literature) for control of software development in cooperative settings, and a set of principles and guidelines for the design of an information infrastructure that provides managers with control information. As implications for research we identify the need to develop new theories of software process improvement through the lens of inter-organisational networks, and to take into account relevant practices from the world of open-source software development. We also discuss lessons for managers of outsourced software development projects. Our results have been validated via expert interviews and by a panel of experts.

Keywords: Software process improvement, measurement, metrics, outsourcing.

1 Introduction

Control through metrics is a critical component of the success of software process improvement programs (SPI) [1]. Much has been written about metrics program implementation in SPI efforts [2, 3, 4, 5]. However, despite the importance of metrics and the presence of this vast body of literature, we have learnt in a five-year study of a large outsourcing organisation that managers experience difficulties in applying metrics in practice. The reason for this seems to be twofold. Firstly, this problem arises not *despite*, but *because of* the presence of a vast body of literature: there are simply too many metrics to choose from. Secondly, outsourcing of software development creates a different context for software process improvement (and thus for metrics): while SPI takes place at the *vendor* side, the ultimate goal is to create value at the *client* side. Therefore, in this paper we aim at selecting metrics that help managers at the vendor side of outsourced projects to develop software that helps clients to meet their business goals.

Which are the concrete problems in the context of outsourcing? Outsourcing of software development by definition introduces distance in the software development process. Firstly, there is organisational distance: the development team is not part of the same organisation as the client. Secondly, often there is physical and cultural distance introduced by offshoring: either the vendor

Published as: L. Ponisio and P. van Eck. Metrics-based control in outsourced software development projects. *IET Software*, 6(5):438–450, 2012. DOI: 10.1049/iet-sen.2011.0199.

is in a different country or continent than the client, or the vendor itself offshores actual development to a subsidiary in a low-cost country. Both types of distance in turn introduce new issues in software development coordination [6]: lack of adequate timely informal interactions, which presents challenges for coordination, communication (and even trust) in outsourcing organisations. Consequently, managers risk loss of control [7] (in the sense of lack of information about what is going on in their projects), especially at the development side.

To address these issues, this study answers the following question: how can managers at the vendor side in outsourcing relations improve control of the software development projects that they are responsible for? To answer this question, we have examined literature to find concrete metrics and implementations, developing a picture of the measurements field through the lens of software process improvement in an outsourcing context. By developing such a picture, we point at implementations in concrete cases in organisations such as IBM and Motorola, we help practitioners to synthesise and reflect on existing work and we contribute to focusing the direction of interest of project managers.

The current paper presents the results of this study in the form of a framework (Section 4) that contains a set of organisational effectiveness measurements and an information infrastructure that collects and distributes principles, lessons learnt and measurement data in software development projects. This framework is distilled from research findings published over the last two decades at the intersection of three different, but related fields: software process improvement (SPI), metrics, and outsourcing. In Section 2, we provide the necessary background information about these three fields by discussing how metrics are used to improve control in outsourcing projects. After that, in Section 3, we present the research method we followed in our literature survey. Our findings are presented in Section 4. Section 5 discusses implications for research and SPI practice. Finally, Section 6 reflects on the results of this paper.

2 How do project managers control outsourcing projects?

The research reported in this paper took place in the context of a 5-year research project [8] in which we observed outsourcing projects at a large outsourcing company that develops software on behalf of customers. Our aim was to discover mechanisms to support managerial decision making during software development processes in these projects. In particular, we were interested in finding how managers deal with software development issues such as complexity and requirements transfer between the customer side and vendor side. By managers we mean project managers at the vendor side who are responsible for realising what the customer ordered. Those people have a variety of titles; we will use 'project manager'. Even though such project managers belong to the vendor side, they are close to the customer side and are in a position to suffer the same lack of control that Lacity et al. [7] indicate as a well-known risk in outsourcing. For instance, in the projects we observed the project manager was physically located in the same country as the customers, while the development team was located in Asia.

In this study we described information paths, made coordination issues explicit and derived guidelines for managing outsourcing projects [9, 8]. We also observed, in line with existing research in the outsourcing domain [10, 11], the importance of building congruent client-developer relationships to deliver within time and budget software products that support the client's business goals, and the importance of gathering enough knowledge to solve the business-technical domain cut-off. Boehm and Sullivan [12] describe a cut-off between the decision criteria that tends to guide software engineers (technical domain) and the value creation criteria of organisations in which software is developed (business domain).

To maximise value creation, project managers must understand the connections between, on the one hand, technical decisions that shape the solution built at the vendor side and, on the other hand, enterprise-level value for the client. 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 easily later if time runs out. Acting as boundary objects [13, 9] between the development team and the customer, project managers in software development strive, thus, to connect technical decisions with value creation criteria (in an attempt to derive the best possible result).

In this context a need appeared over and over again: project managers, sensing lack of control, asked for effective mechanisms to better control software projects, *i.e.,* mechanisms that help them steer the project such that decisions are aligned across strategic, technical and organisational domains. *"I need to see what's going on at the other [the development] side."* and *"Metrics could help us to increase control"* were phrases that came up in our interviews. After asking *"how are we in control?"* [8] project managers asked how they could improve control. They were referring to the development projects they execute and are responsible for, and from which they have to realise customer value. This leads us to the following objective for this paper: to find a concrete set of metrics that help project managers to improve control of their software development projects. Thus, we are looking for mechanisms project managers can use to gain knowledge about their projects, for instance about coordination and customer value.

2.1 SPI, metrics, and management of IT outsourcing

Metrics have the potential to improve software development processes but the field is heavily populated. There are too many metrics to choose from and project managers do not have time to select appropriate metrics for their projects. Even though the SPI literature is rich in examples of metrics applications and management of IT outsourcing, projects present specific problems that need to be addressed.

2.1.1 SPI

The fundamental objective of SPI is to change software development processes in order to achieve improvements in quality and productivity [14]. The SPI literature is rich in examples of metrics and in principles to apply them [15, 16, 17, 4, 18, 19, 20]. Already in 1999, Rico had identified more than 487 metrics for software process improvement [21]. By 2004, Hansen et al. had considered 322 works on software improvement [22]. Since then, the body of literature has only grown, creating a need for integration to support practitioners.

Firstly, several researchers have attempted to come to a more *integrated body of literature* of SPI [23, 24, 25]. Several papers present frameworks or critical evaluations [22, 26]. For instance, in a literature review of contributions to the SPI field, Hansen et al. created a framework categorising contributions as prescriptive, descriptive, or reflective [22]. The authors conclude that the field is heavily biased towards contributions that tell software professionals how they can carry out software process improvement initiatives (prescriptive contributions). According to this study, much less contributions either report on actual instances of SPI programs in software organisations (descriptive contributions), or set the other contributions in a theoretical context (reflective contributions). In our search for related work, we focussed on descriptive contributions.

Aaen *et al.* made a survey of the SPI literature and experiences from SPI practice [14]. Their paper offers a conceptual map of nine key ideas underlying SPI organized in three concerns: management, approaches, and perspectives of the SPI process. Each concern includes ideas associated to it. For instance: organisation, plan and feedback are the SPI ideas corresponding to the concern of management; evolution, norm and commitment are SPI ideas corresponding to approaches; and process, competence and context are the SPI ideas corresponding to perspectives in the SPI process.

Florac, Park and Carleton [4] define processes for selecting and defining metrics for process management and improvement. Besides offering a list of technologies and methodologies for changing software processes, they list measurable entities (for instance, people, tools and procedures) and

their attributes (such as experience, accessibility and coverage, respectively) that can be measured to address, for example, process compliance (compliance measures help managers to understand why a process might not be performing as it should and performance measures address the degree to which a process fulfils its purpose).

Secondly, a number of papers focus on reporting experiences with *applications of metrics* in organisations. For instance, early research described cases where metrics were applied in practise in leading companies such as Motorola [27] and AT&T Bell Laboratories [28]. More recently, Iversen and Mathiassen presented a case study that analysed an engineering process in which a metrics program was constructed and put into use [5]. The program's goal was to test the effect of ongoing SPI initiatives within the company Danske Data. These works, and those of the next paragraph, are closely related to our literature review, which we present in Section 3.

Thirdly, other papers focussed on studying factors that influence success of SPI programs [29, 30, 31], and (in line with research in the area of software metrics) there is a body of work that emphasizes the need to apply metrics *in accordance with* company-specific needs. In particular, Iversen and Kautz [32] and Kautz [33] assert the importance of adapting metrics programs: to be successful, the metrics programs implemented should be defined according to the organisation's specific information needs. Furthermore, while describing examples from practice, Aaen *et al.* [14] illustrate that there is room to implement SPI programs in very different ways and that metrics must be adapted, at the time of implementing them, to the specifics of an organisational environment.

2.1.2 Metrics

The benefits of using metrics in software development are unquestionable: the use of information provided by metrics in decision making leads to higher organizational performance [31] and visible results (such as those provided by measurements) are considered critical to success of any improvement plan, keeping participants focussed and motivated [1]. However, whilst the literature recognizes metrics are an important source of control in software development, project managers still ask *"which metrics can I use in my project?"*, indicating they have time neither to research the best metrics nor to find the best advice to apply them.

There are many metrics, but equally important as the metric is its application. The major problems of metrics are using measurements in isolation, handling uncertainty, combining them with evidence, and gathering and applying measurements that are meaningful [14]. Having numbers to show does not mean per-se neither that the measurements are relevant and meaningful, nor that they are accurate and reliable. Size is a case in point: "One single measure of size could give a misleading picture of progress and cost" [2].

2.1.3 Management of IT outsourcing projects

There is a vast body of literature on IT outsourcing. Lacity, Khan and Willcocks [7] review the IT sourcing literature and present, among other results, an overview of determinants of IT outsourcing success. They distinguish between three categories of determinants: determinants related to how the decision to outsource was made, determinants related to characteristics of the outsourcing contract, and determinants related to controlling the relation between customer and vendor. This last category includes issues such as trust, communication, information exchange, and cooperation, which makes explicit that outsourcing involves issues related to the customer-vendor relationship [34], socialisation in global software development [35] and coordination [10, 36]. For our research, this category is the most relevant.

Also the issue of quality is related to what Lacity et al. call 'relationship governance'. Van Ekris [37] claims that not only quality of the final product is important: low customer perception of 'delivery quality' (i.e., quality of the process of delivering that final product in a relationship with the customer) may rule out a supplier for the next project. Thus, a significant problem (of outsourcing) that metrics need to address is the need for effective mechanisms to help managers

to establish a connection between enterprise-level value maximisation (which determines product and delivery quality for the customer and is thus in the business domain) and technical decisions (domain of technology, mostly at the side of the vendor) [9].

Gopal and Gosain hypothesised that higher levels of quality-based outcome control are associated with higher levels of software quality [38]. In an empirical study of 96 projects, the results indicate a significant and positive effect of control on quality. According to that study, activities such as knowledge sharing between the client and vendor teams increased quality because, intending to eliminate gaps in understanding requirements and ensuring clarity of expectations, these activities diminished the cut-off between the decision criteria that tends to guide software engineers and the value creation criteria of organisations in which software is developed [12]. In other words, they bridged the gap between business and IT domains. Their paper does not address the question of whether and how each of these projects itself chose a set of metrics for control. This question has been addressed by Misra who proposed a top-down framework for selecting outsourcing metrics [39]. Misra observes that "The whole process of analyzing metrics and selecting them for a particular outsourcing project can be very tedious and time consuming" and proceeds to describe best practices.

All in all, we observe that many concepts and best practices have been proposed. Yet, a major problem remains: it is unclear which metrics to use. In spite of significant research efforts, there is a need for practical findings that help practitioners to know when, how and where to measure, what not to measure to control their projects, and all that in a nutshell.

3 Research Method

The research reported in this paper took place in the context of a long-term interpretive research project at a large outsourcing company in the area of IT that we name BIG. BIG is a good example of an IT service provider because it exemplifies most (large) offshore outsourcing development organisations. BIG has offices in many countries, with headquarters in the UK. 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 methodological approach adopted in our research at BIT follows the guidelines of Klein and Myers [40] and Walsham [41, 42] on methods for carrying out interpretive case studies. We decided to adopt an interpretive approach because the close involvement it requires increases relevance to practice [42], which is one of the points of this particular work. Klein and Myers [40] demonstrate that case study research can be interpretive and indicate seven principles of interpretive research, which we followed in our case study. According to Klein and Myers, 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 sensemaking 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.

In our research at BIG, we were interested in finding how managers deal with software development issues such as complexity and requirements transfer across domains. The objective was to understand mechanisms that organisations put in place to optimise software development management in outsourcing projects. In particular, we studied documentation of such projects, such as the requirements management plan, and best practices used at BIG. We obtained knowledge about which mechanisms facilitate transfer of the information that managers need when making decisions during software development [8].

At BIG, we observed four large offshore outsourcing projects and analysed any mechanism we could observe. The four projects we observed at BIG represent 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. The observations took place between January 2007 and March 2011 and were made by two researchers. To double-check the findings and to detect potential misunderstandings, focus groups and extra interviews with two experts were performed. The experts were software architects with more than ten years of experience each in managing software development projects.

One iteration in this hermeneutics cycle [40] that spans five years of interpretive research at BIG called for the research we present in this paper. By interpreting our observations we realised that practitioners at BIG expressed a need for a framework that empowers them to see what is going on in their projects, in order to improve their control over these projects. In particular, the need to improve control in outsourcing projects (in the sense of obtaining knowledge about the product and the coordination) was mentioned in two interviews we held with project managers (experts in software development outsourcing). Software metrics seemed to be the logical approach to improve software development in this case because when used appropriately they can show what's going on.

3.1 Research design

As stated above, our five-year-long case study at BIG indicated the need for research to elucidate a way to improve control in the projects of this outsourcing company by a compact set of metrics. In this particular iteration of the hermeneutic cycle, we followed the guidelines for design science as described by Hevner et al. [43]. This design science approach fits our research because our long-term research calls for elucidating a small set of metrics that helped project managers to improve control in outsourcing projects; we view this set as a design artefact. This artefact is a framework built upon research literature: we used existing research that describes metrics applied to real-world projects. We then designed our framework by filtering the papers according to our experience in programming and software development processes and our observations at BIG. Using this process, we follow the guideline of Hevner et al. to treat design as a search process.

To evaluate usefulness our design artefact (another guideline of Hevner et al.), we seek feedback via expert interviews, as reported in Section 4. In particular, we seek expert opinion about the question whether our results are useful and applicable in other organizations than those that report their experiences in the works we have selected.

As the framework we designed and present in this paper is based on research literature, our research design roughly follows a general procedure for performing systematic reviews described by Kitchenham [44, 45], even though we do not view our own result as a systematic literature review. By following Kitchenham's guidelines, we strive to make our research as repeatable as possible (although it will not be completely repeatable as human judgement is involved in interpreting the usefulness of the metrics found).

3.1.1 Research questions

The problem that we introduced in the previous two sections leads to the following question that we want to answer by our literature search: which software metrics are reported in the software process improvement literature that are applicable to software development in an outsourcing context and what practical experiences with metrics have been reported?

3.1.2 Search process

As a starting point, we took the entire set of papers of the most recent EuroSPI conference at the time of writing, which was EuroSPI 2010. We then added all papers referenced in the papers in this initial set, followed by all papers referenced in the newly added papers, until no more new papers were found. As EuroSPI is a quite focused conference, this process turned out to be feasible.

3.1.3 Study selection process

Based on a close reading of the abstract, title and keywords (if available), we selected those papers that appear to present definitions of concrete software metrics applicable in an outsourcing context or experiences in applying those metrics in a practical situation. In other words, we selected those papers that present either concrete metrics or experiences with metrics in organisations, or by addressing elements to be measured (for example, aspects of open-source software development) that are relevant for the perspective on metrics that we discussed in the previous section. Since we needed our metrics to be feasible, we included articles that provided evidence of the feasibility of implementing the metrics in outsourcing organisations such as BIG.

3.1.4 Quality assessment

Most of the papers in our selection are themselves in the category of design science. Therefore, to assess quality of each paper, we checked whether it complied with the 'Research contribution' guideline of Hevner et al. : does the paper provide a clear and verifiable contribution?

3.1.5 Data extraction process

From the papers deemed applicable, we extracted those metrics that, given our experience as developers, we would like to have if we were project managers. We did not extract metrics that are specifically for outsourcing only, as doing so would limit our view of the problem and potential solutions. We focused on the metrics that had some potential to describe projects in the cooperative way to develop software today. Since the research reported in this paper was conducted in the context of an interpretative study, observation and analysis based on our experience as software developers made us gain knowledge of which contributions would fit our needs. We used that knowledge to cross-check the information gathered in the interviews with the results of our literature review. The role of researchers as decision tools using their experience to filter the papers acts both as an enabler of this framework (as selecting a compact set of metrics is the key value of our contribution) and as an unavoidable limitation.

The result of our literature review gave 959 metrics. We organised this large amount of papers by applying an existing framework for organisational effectiveness measures, as explained in Section 4. As a result we obtained a tree of categories that we think is useful for practitioners to find the metric they need. Moreover, this organisation helped us to check whether there was at least one metric per category. Given that synthesis was paramount for the framework we could suggest to practitioners: our results are digestible and systematic.

3.2 Study limitations

By following the process described above, we incur the risk that the list of papers collected by the process described above is not complete, for two reasons. Firstly, it is possible that relevant papers exist that are not referenced in any EuroSPI paper nor in any paper referenced by EuroSPI, nor in any paper referenced by papers referenced by EuroSPI papers, and so on. (In other words, relevant papers from a completely disconnected research community). To account for this risk, we cross-checked the list with our own database of papers on SPI, outsourcing and metrics, which was collected in over nine years of research on these topics. Secondly, there is a risk that we excluded papers from the list because we misinterpreted the abstract (or the abstract provided insufficient information for a good verdict). We have indications that this risk is not large: we received several suggestions for additional literature from reviewers of an earlier version of this paper, all of which were included in our original data set.

Our framework is not the presentation of the outcome of a systematic literature view, but a design artefact that is based on a literature review. While the risk that our search process was not complete was mitigated as much as possible by the cross-checking process described above, note

that the resulting framework presented in this paper is not complete in the sense of Kitchemham's requirements for systematic literature reviews. This makes sense because, according to Kitchenham, a systematic literature study per definition aims to present all papers, while the framework we needed to design calls for quality (*i.e.,* metrics that are suitable for improving control), which in this case is the opposite of completeness: we required a first compact set of sensible, practical metrics.

4 Research findings: a metrics framework designed for outsourcing

We present our research findings in a framework that consists of two parts: a set of *organisational effectiveness measurements* (Section 4.1) and a set of *information infrastructure principles* (Section 4.2). Organisational effectiveness measurements are software metrics in a broader context, as we explain below. The information infrastructure principles are the starting points for the design of a system of components that provides managers with information to control software processes.

4.1 Organisational effectiveness measurements

This section reports software development measurements for SPI found in the existing literature. We selected these metrics using the method we described in Section 3, bringing together relevant proposals for companies in an outsourcing context. Having a manageable list of metrics and advice, projects managers can better understand how they can put metrics to use. Our selection of metrics is presented in Figure 1.

Why do we use the term 'organisational effectiveness measurements' instead of software metrics? As we have argued before, software metrics need to be applied in their organisational context, which, in current practice, often means globally distributed cooperative software development. Exploiting the notion of *organisational effectiveness* as proposed by Applegate [46], we systematically identify the organizational context for software metrics.

Applegate's notion of organisational effectiveness suits our control-for-more-value needs because it sees control as enabled by shared understanding of relationships between strategy as executed and organisational effectiveness. In more practical terms, it can be used to design what to measure if our goal is to gather information upon which to base management decisions.

Thus, in our framework the metrics are organised in categories based on Applegate's view of important characteristics for organisations in this information age. The four areas of interest in measuring organisational effectiveness measurements (adapted from Applegate [46]) are (i) *results*, which are needed to know how the software quality assurance process is performing, (ii) *stakeholder satisfaction*, (iii) *industry dynamics*, and (iv) *software process performance*, the set of "activities, methods and transformations that people use to develop and maintain software and the associated products, for example: product plans, design documents, code, test cases and user manuals" (SEI).

Why do we select these metrics rather than others? The software metrics that comprise our set of organisational effectiveness measurements are metrics that help managers to control software development projects. We are specifically interested in metrics that project managers can use to control software development performed in a cooperative context, such as outsourcing. Control in this context is the ability to develop an understanding of what is going on in the project and make informed decisions. Moreover, our selection comprises metrics that, according to existing literature, have been tried in real projects of real organisations, as is indicated by the company names in the column labelled 'Organisations'. (In this column, the name 'SEI Capability Maturity Model for Software' refers to organisations that have implemented that model.). By using Applegate's categories, this work coincides with the results of Iversen and Ngwenyama, who were the first to instantiate Applegate's framework for the SPI domain [1].

We illustrate the use of our framework with the following example. Consider a manager of an offshored outsourcing project who wants to increase control. A significant part of having that control is to know exactly how work is distributed in the offshore team: Who is who in the offshore

Organisational effectiveness metrics				
		Metric	**Organisation**	**Reference**
Results	**Size**	Functional size	Hewlett Packard, Eclipse, Danske Data	Grady97, Iversen00
	Code ownership	Developer participation: how is work distributed? E.g., can we recognise "partitioning" in the code?	Apache server	Mockus00
	Time	Adherence to schedule: variation from agreed time of delivery, absolute and relative to volume of project.	Danske Data	Iversen00, 03 and 06
	Cost	Adherence to budget: variation from estimated use of resources.	Danske Data	Iversen00, 03 and 06
	Quality	Number of error reports relative to size in function points.	Danske Data	Iversen00, 03 and 06
		Number of error reports, absolute.	Danske Data	Iversen00, 03 and 06
		Defect density	AT&T, Motorola, IBM, Apache server, Eclipse, Danske Data	Barnard94, Rosenberg94, Florac97, Mockus00
		Defects per line of documentation	Hewlett Packard	Grady86
		Defects per testing time	Hewlett Packard	Grady86
		Defects per thousands of non-commented source statements	Hewlett Packard	Grady86
		Post-release defects per thousand lines of code added, instead of delivered.	Apache Server	Mockus00
		Non-commented source statements per engineering month	Hewlett Packard	Grady86
		Cyclomatic Complexity	IBM Rochester SPI	Kan95, McLoughlin10
		System paritioning	IBM Rochester SPI	Kan95
		Fan in	IBM Rochester SPI	Kan95, Florac97
		Fan out	IBM Rochester SPI	Kan95, Florac97
Stakeholder satisfaction	**Customer satisfaction**	Survey in-person, phone and mail, random, systematic and stratified.	IBM Rochester SPI	Kan95
		Satisfaction with the development process (questionnaire)	Data Dansk	Iversen00, 03, and 06
	Employee satisfaction	Satisfaction with the development process (questionnaire)	Data Dansk	Iversen00, 03, and 06
Industy Dynamics	**Market performance**	Any lost bids	Not applicable or unreported	McLoughlin10
		Loss of reputation to the firm	Not applicable or unreported	McLoughlin10
		Satisfaction with the development process (questionnaire)	Data Dansk	Iversen00, 03, and 06
Software process performance	**Productivity**	Number of individuals submitting reports (eg. bugs)	Apache server	Mockus00
		Size of the development community	Apache server	Mockus00
		Core team size	Apache server	Mockus00
		Cumulative distribution of contributions to the code base	Apache server	Mockus00
		Resources used to develop the system relative to volume of project hours (hours/FP).	Data Dansk	Iversen00, 03 and 06
		Gain per Year in Productivity	SEI Capability Maturity Model for Software	Herbsleb94
		Gain per Year in Early Detection of Defects	SEI Capability Maturity Model for Software	Herbsleb94
		Average fixed defects per working day	Hewlett Packard	Grady86
	Time	Time used in review meetings	Small company	Iversen00
		Reduction per Year in Calendar Time to Develop Software System	SEI Capability Maturity Model for Software	Herbsleb94
		Time to resolve problem reports	Apache server	Mockus00
		Percent overtime per 40 hours per week	Hewlett Packard	Grady86
	Cost	Resources used in coordination activities	Small company	Iversen00
		Thousands of Dollars per Year Spent on SPI	SEI Capability Maturity Model for Software	Herbsleb94
		Dollars per Software Engineer per Year Spent on SPI	SEI Capability Maturity Model for Software	Herbsleb94
		Average engineering hours per fixed defect	Hewlett Packard	Grady86
	Quality	Reduction per Year in Post-Release Defect Reports	SEI Capability Maturity Model for Software	Herbsleb94
		Business Value Ratio of SPI Efforts	SEI Capability Maturity Model for Software	Herbsleb94
		Average fixed defects per working day	Hewlett Packard	Grady86
		Perception of delivery quality	Large consultancy company	VanEkris08

Fig. 1: Organisational effectiveness measurements.

Figure label	Reference	Figure label	Reference	Figure label	Reference
Barnard94	[28]	Herbsleb94	[19]	McLoughlin10	[48]
Dekkers99	[20]	Iversen00	[32]	Mockus00	[47]
Florac97	[4]	Iversen03	[5]	Rifkin91	[16]
Grady86	[15]	Iversen06	[1]	vanEkris08	[37]
Grady97	[17]	Kan95	[49]		

Tab. 1: Cross-reference of figure labels and the literature references of this paper.

team? How is code authorship distributed? Is most of the coding done by one single developer, or is it partitioned in equal shares? In our framework, one of the metrics characterising a software artefact is *code ownership* (second metric under *Results* in Figure 1). This metric indicates developer participation. It has been successfully used in the Apache server project and we can get the details of that experience from an article written by Mockus, Fielding and Herbsleb [47], indicated under *Reference*, the third column in our framework (Table 1 provides the key for the references).

4.2 Information infrastructure

As stated before, we are interested in metrics that project managers can use to control outsourced software development. According to general models of control, a controlling system (in this case: a project manager) needs information about the system that it tries to control (in this case: a software development project in a cooperative context). The software metrics literature discusses the many different metrics identified in the software field that can serve as control information, and that we have presented in the previous section. This information, however, needs to be made available to the controlling system. The *information infrastructure* is the system that connects the controlled system to the controlling system and supplies the controlling system with information.

In Figure 2 and its continuation (Figure 3), we present a set of principles that can serve as a starting point for the design of such an information infrastructure. The table describes principles organised in four categories: information management, information access, communication management and presentation. A principle is a piece of advice regarding the way to apply metrics; it is backed up by pieces of knowledge that some manager gained when implementing metrics programs. In other words, the principles are taken from actual experiences in SPI measurements program application described in the research literature. We have selected those principles that, based on experiences of applying SPI programs, provide managers with information to control software processes.

As a first example, consider the principle *Plan to throw one away* (sixth principle in category *Information management* in Figure 2). Our framework warns project managers not to make the result of a measurement effort completely dependent on too few metrics, and to take into account that some metrics will not be accurate enough, or even collected. As indicated in the rightmost column of Figure 2 and Figure 3, we have found a recent article from Iversen and Ngwenyama [1] describing SPI implementation in a company, where the experts could not include as many metrics in their report as they had planned. The principle *plan to throw one away* has a description of the principle (third column), what not to do (fourth column), experience in an actual SPI program in the form of an example or quote in the fifth column (in this case, *"The first measurement report included only 20% of the projects and only three out of six factors"*), and a literature reference (sixth column, in this case "Iversen06") for the manager who wants to know the details.

Our second example warns project managers not to make the mistake of underestimating the cost of measurement (they usually add no extra time for measurement in the project plan). Moreover, they might have thought that to compute function points automatically was one way to reduce the burden of extra work for their people. Indeed, at first sight automatically computing function points seems to be the clever thing to do, especially in projects with very limited resources.

Information infrastructure (part 1)					
		Principles	**What *not* to do**	**Quote / Example**	**Reference**
Information management	**Start by determining goals**	Define clear outcomes to expect and collect the data based on clear objectives. An example of a clear outcomes to expect is ``to let all developers work on all parts of the product". Using Basili's Goal Question Metric method is a good way to design the metrics to include in the program, being based on the goal that stakeholders expect to achieve from the metrics program.	To implement a set of metrics that is not well suited to describe a concrete outcome. An example of a too general objective is ``to give information about the effect of improvement initiatives".	``*Improved procedures for documentation of source code should allow all developers to work in all areas of the companies products and should facilitate the extension of the development teams.*"	Iversen00
	Match your goals with measurable attributes	To have a clear goal such as improve efficiency by x% is important, but do neither procrastinate nor forget to choose the attribute to be measured to check if the goal is met. It should be clear to management that the chosen attributes to measure (and the metrics results) describe the program's goal.	To underestimate the importance of having process and product attributes that describe the goal, leaving the decision of what should be measured to the group responsible of the metric program.	``*We expect to gain a 10% improvement in efficiency has become an important focal point of the SPI project [...] However, neither the CEO nor the contract was explicit on what should be measured to show this 10% efficiency improvement.*"	Iversen03
	Establish incentive structures	Metrics programs seem to be more successful if people see that they bring improvements to the process or the product. It should be clear: 1) what data to report, 2) how to report that data, 3) why data they provide is important, 4) show results based on the data.	To not inform project managers about exactly what data they should report, how they have to report it, what part of the process and product will be better by analysing that data.	``*Those who report data to a metrics program need to see some form of advantage in the program.*"	Iversen00
	Establish a project	A metrics program consume resources and therefore to set it in the context of a project for its own sake should make the task of collecting and analysing the data easier. Moreover, you need the right staffing to carry on a measurement program.	To have people work on metrics as an extra task of their current projects, forgetting to recognise that collecting and reporting metrics consume resources (e.g., time).	``*Establishing a formal project made the program far more visible in the organisation and made it much easier for the participants to argue that adequate resources should be available.*"	Dekkers99 Iversen00 Iversen06
	Start simple	In the beginning, collect a small set of goal-oriented metrics. For example, ``One company measured the number of fixed change requests delivered on time and the time used in review meetings and found that change requests delivery on time had increased from 45% to 77% and review meeting time was shortened by a factor 4."	Neither to start with a large metrics set, nor to start with too general metrics. Too many resources will be spent on them and people will not see accordingly advantages when the (analysis report containing the) results comes back to them.	``*six fairly complex factors should be measured, and all projects were required to report data from day 1. This was an extremely ambitious undertaking, and as of yet, all the factors have not been measured, and some have even been officially abandoned.*"	Iversen00
	Plan to throw one away	Some measurements will prove to be too difficult to get due to wrong initial assumptions and inaccuracy in the data access.	To underestimate measurement. For instance, planning that the measurement will be made in a completely automatic way.	``*The first measurement report only included 20% of the projects and only three out of six factors.*"	Iversen06, Rifkin91
	Use organisational knowledge	Metrics programs must take into account the existing work practices in the organisation, and the needs of the stakeholders affected by a potential effect of the metrics application.	To not communicate clearly the advantages expected from implementing the program.	``*external consultants acted as analysts of the current practice and carried out interviews with the developers [...] This provided the knowledge necessary to define metrics and to gather data. *"	Iversen00
	Consider potential problems when measuring size	Measure size accurately is critical because size is a key attribute to measure common goals such as efficiency and productivity.	To believe blindly the first result obtained from counting function points wihout checking that results match goal and match questions, i.e., checking that numbers obtained match the perceived size of the system.	``*Excluding a size measure seriously impeded reaching the original objective of measuring efficiency and productivity, as there was no longer a measure of the output of the software projects.*"	Iversen06
	Match measurement with your organisation's goal	Attributes to measure, measurement data and its results need to be recognised by management. Measurement must describe part of the project in the eyes of stakeholders. Otherwise measurements become unacceptable.	To see measurements as just gathering data without matching them to a business goal. For instance, to use function points count without normalising them when needed.	``*...after counting function points in several application systems, it was very difficult to see any relationship between the perceived complexity of the systems, and the number that the counting procedures had arrived at.*"	Iversen06
	Have a complementary suite of measurements	The measurements you choose should be complementary. Each measurement should contribute to improve the picture of the system obtained from the measurement program.	For instance, to count code performed during the original development twice.	``*Enhancement projects that continued work on an existing system were accredited the entire function point count of each module they modified, even if the modifications were miniscule. This gave very few hours per function point, or in other words, unrealistically high productivity.*"	Iversen06 Dekkers99
Information access	**Use improvement knowledge**	Implementing metrics programs involves several branches of knowledge such as SPI, software development and reverse engineering. One solution might be to include external consultants.	Disregard unfamiliar areas of knowledge (for instance, reverse engineering if management has background in software architecture) when making decisions about how to implement the metrics program.	``*The first measurement report [...] was criticized for being too academic.*"	Iversen00
	Use non-invasive measurements whenever possible	Facilitate collecting data by making it simple. Metrics from finished projects would be used as a baseline and metrics of finished parts of projects (when collecting them makes no harm) should be used when possible. Beware that some stakeholders will not provide data.	To have tedious mechanisms to report the data, especially with unclear questions.	``*There are some who simply do not enter data into the system. There are some that have misunderstood the definition of the field.*" ``*...results from 13 out of 56 projects that were completed*"	Iversen03
Communication management	**Publish Objectives and collected data widely**	People need to see that metrics they collected are used and that bring some advantage. Publishing realistic objectives is a way to secure gathering data of quality for the SPI program, and to improve the metrics program with employee's feedback.	To be vague in the objectives and to relate the metrics results to performance evaluations. The objective of using metrics is to improve the way we do things rather than to find who is to blame. Not informing the results of the metrics program might form undesirable rumours.	*(About why developers did not enter data)* `` *this information was not previously used for anything*" ``*Data discipline was improved in the next report .*"	Iversen00 Iversen06
	Facilitate debate	Not only communicate the metrics, but incentivise stakeholders to discuss the metrics program and its results. Use their feedback to improve the measurement program	To not hear what employees have to say about the metrics program and its implementation. This might cause the loss of valuable improvements for the program (such as improving the input fields of the system used to report data).	``*Another problem was that questionnaires [..] covered questions relating to contractual agreements and to the entire course of the project, whereas those who answered the questionnaire were users who were only involved in acceptance tests.*"	Iversen00

Fig. 2: Information infrastructure: principles for the design of a system that provides managers with information to control software processes (part 1).

Information infrastructure (part 2)					
		Principles	**What *not* to do**	**Quote / Example**	**Reference**
Communication management	**Use the measurement feedback**	Empower the program with the feedback of the employees that develop and collect measurement data.	To enter in this vicious circle: ``*the data reported were of a poor quality, since those who reported them did not see any advantage in supplying accurate data in a timely manner. At the same time, the poor data quality caused management to be wary of making the results public.*"	`` *being able to recognise trends, even from imprecise or non-complete data sets, can be more helpful than having no data at all.*"	Iversen00
Presentation	**Facilitate feedback with good layouts**	Feedback empowers the measurement program. One of the mechanisms to facilitate feedback is to use the information gathered, for instance by emitting reports containing the results of the measurement program.	To have forms to fill in the measurement data that are difficult to understand. We don't have examples of 'bad' layouts, but examples of good ones can be found in Florac (1997)	*(about why developers did not enter data)* ``*the user interface for the application used to enter the data was highly confusing, giving rise to many wrong entries.*"	Iversen00 Iversen06

Fig. 3: Information infrastructure: principles for the design of a system that provides managers with information to control software processes (part 2).

Our framework reminds project managers of experiences reported in SPI literature that warn of one danger with this approach: inaccurate results. Look at Figure 2, *Match measurement with your organisation's goals* in the category Information management. The Quote/Example column in our framework points out an example where after using function points, managers could not see any relation between the perceived complexity of the system and the obtained metric results. Moreover, by following the reference that our framework offers next to it ("Iversen06"), the reader can learn about the reasons behind this misalignment between function points and goals. This challenge could be explained by a known problem with function points: maintenance projects being credited the entire function point count of each module they modify (even for tiny modifications), thus indicating unrealistically high productivity. In this way, using our framework, managers are warned of the risk of automatizing function point counting.

4.3 Evaluation of the proposed framework

We have validated our framework via two interviews with a project manager from BIG with 10 years of experience in managing outsourcing projects and with a panel of experts. Both evaluated the framework as useful to improve control. The project manager suggested that our framework could help managers mitigate the risk of loss of control over their projects. The responses of the experts indicated they found the framework very useful because (subsets of independent) metrics from our framework could be used to support mitigating several risks such as introducing requirements defects (thanks to better feedback and to driving the focus of attention to internal dependencies), reducing commercial risk (*e.g.,* loss of revenue due to misconfigured interfaces with other systems), and by our metrics and principles helping managers to balance time, functionality (including quality) and cost.

The opinion of the outsourcing manager Reflecting on the potential of the framework applied to general situations, our expert could relate to this approach, finding it useful to improve control in outsourced projects. In particular, the interviewee was very enthusiastic about the guidelines and principles, recognising especially the principle *Plan to throw one away* and strongly agreeing with the guideline that suggest to analyse cohesion and coupling (which we present in Section 5.3).

Is our framework suited to address the outsourcing measurement challenges? The expert agreed that our framework would be useful in the current outsourcing context because it helps companies to meet the current outsourcing challenges such as lack of time, need for semi-automatisation and competitiveness. In the opinion of this expert, our framework

> *"is OK because metrics are important, but there is no time to report findings or searching for the best metrics: it is all about finishing a project on time. We would need some*

> *kind of automated system, so that there are fewer things to do."* (outsourcing manager at BIG)

Surprisingly, our interviewee was concerned about the framework's potential to replace managers. This was never the authors' intention, who consider the thought of replacing people by frameworks is totally unrealistic. The idea behind our framework is to empower managers by giving them the right information.

The opinion of the panel of experts We consulted five outsourcing experts from the Dutch outsourcing industry (two from an organisation in a client role, and three from organisations in a vendor role). The experts indicated that they found our framework very useful:

> *"There are enough metrics and principles that are usable and that can add value in an outsourcing relationship."* (expert from organisation in client role)

The experts expressed the idea of using our framework to pick some metrics to improve understanding of their project and agreed with the principle (stated in our framework) of starting small:

> *"To implement all would be 'extreme'."* (same expert)

According to one of the vendor experts, the metrics can be used to mitigate commercial risks by preventing defects due to misinterpretations (Lauesen and Vinter [50] call this 'requirements defects') and are due to tacit requirements. This expert based his view on his experience from a large migration project planned to last three to four years.

Furthermore, a specific positive point of the framework is that it drives the focus of attention to inter-dependencies; which the experts can use to improve coordination. One of the experts, who belongs to a large organisation that combines custom software together with commercial-off-the-shelf suites, saw the advantage of using some of the metrics in our framework to improve the efficiency of configurations with interfacing suites.

Another possibility of using our framework (mentioned by another expert) is to check whether a vendor is capable of building the product that this vendor promises in a bid. This is because the company of this expert uses productivity index and manpower built-up index together with lines-of-code to check whether plans offered by a vendor are realistic. Although not mentioned by the experts, we believe that the same strategy could be used by a vendor in a bid to convince the customer that their bid is realistic.

According to the experts, our framework could help them in some specific tasks such as improving tracking progress of the project, improving provision of feedback to the development team and improving management of software defects. Moreover, in general, the set of metrics and principles can be used to balance time, cost and functionality.

Two experts raised the issue of trust in the customer-vendor relationship. Although recognising the challenges involved in quantifying trust, they stressed the need for some mechanism to be able to measure and control trust. At BIG, such mechanisms were already in place. Categorisation of customer-vendor relationships according to their types and guidelines in internal documents help managers improve trust in the customer-vendor relationship. This explains why trust is missing in our framework; we agree with the experts that mechanisms to control trust need to be added if not already covered by other mechanisms.

Generalizability Can our metrics be of use for companies other than the companies mentioned in the referenced articles such as Motorola or AT&T? This is something we cannot prove, but the results of the interviews are encouraging at the very least, given that the experts we consulted were enthusiastic about their potential. Moreover, metrics are good or bad according to a company and project-specific needs. However, we do believe our framework is potentially useful in other companies because it includes metrics that are recognised, as indicated by our experts, by highlighting issues of outsourcing projects such as modularity and cooperation needs.

5 Implications for research and practice

This paper contributes to the growing understanding of how to improve control in software development projects. Our work extends the existing literature by bringing together relevant proposals from the SPI literature to build a framework that is specifically suitable for vendors in an outsourcing context. The outcome of our search for metrics (and practical experience with them) reported in the literature is presented in two tables in our framework. These results indicate the following implications for research and practice.

5.1 Enriching the literature with models that support feedback between the operational and the strategic level

The metrics that we present in Figure 1 all deliver measurements data of the ongoing current operations of an organisation: They are at what we call the data or operational level. Metrics have been recognised as a way to connect this operational level to the strategic goals of that organisation [51], which are on what we call the managerial or strategic level. For instance, we could use code ownership metrics to support or reject a hypothesis that suggests downsizing by applying a certain personnel cut.

As our first implication for SPI research, we stress the importance of enriching the current literature with models that enable this feedback between the data (operational) level and the managerial (strategic) level. Such models are needed in outsourced software development to support collaboration to deal with dynamism and complexity. Already in 1994, Applegate [46] noted that collaboration is an essential capability for organisations to face dynamism and complexity. Since then, dynamism and complexity have become much more important, mostly because of globalisation and the emergence of the Internet.

Secondly, we believe that on the grounds of current demands for faster reaction in boundary-spanning outsourcing, organisations improving software development need more than ever not only a model that enables feedback between goals and metrics, but a model that supports some degree of *automatisation* of providing this feedback. Semi-automatic models should enable shared understanding of links between organisational strategy and development effectiveness on a regular and timely basis. We realise that while the literature recognises the cyclical dependency between use of metrics in decision-making and organisational performance to influence success [31], the *dynamics* of incorporating semi-automatic models that would enact an increase in that interdependence are mainly unexplored.

Our results agree with these insights, since increased control without sacrificing flexibility can be achieved by timely feedback between development performance and organisational strategy. The set of metrics and principles in our tables all support this feedback. This sets the stage for further research in the dynamics that result from providing such feedback.

5.2 Towards a focus on inter-organizational networks in SPI research

In the previous subsection, we have discussed the need to address (automated support for) feedback between metrics and goals and to enrich the SPI literature with models that explain the dynamics arising from providing such feedback. In this section, we discuss another direction for SPI research that is indicated by our research: to study software process improvement through the lens of inter-organisational systems. Other people have studied strategic alliances in outsourcing from the business perspective [52]. We focus on software process improvement and propose to extend that research area with an inter-organisational network approach. Our suggestion follows from our reflection on how metrics are used in practice, which indicates that IT managers need to shift their attention to more collaboration-oriented metrics.

To elaborate on this, we start with the observation that the job of project managers is shifting towards more involvement with and increased knowledge of the business. Project managers,

architects, and even developers need to understand the business that IT is supposed to support in order to be successful. For instance, if the system is designed to support electronic customs and e-commerce, then the architect needs to know about areas relevant for this business such as imports, exports, transits, inspections, rates and tax rules. If the system is to support portable sound-intensity measurement, then the architect needs to know about sound power per unit area, threshold of hearing and intensity of decibels. This shift, in our opinion, is reflected by current practice, where the existence of many tools that cover the need for accurate project estimations and progress monitoring makes 'mechanical' work less needed. Our results indicate project managers show now more and more interest in project solutions at a strategic level, such as business alignment benchmark assessment.

In itself, this situation potentially applies to all types of software development. In the case of outsourcing projects, i.e. software development projects performed in one organisation for a customer belonging to another organisation, the required business knowledge for project managers is not only knowledge about the business of their own organisation but also that of their customers. In other words, in this case software development is performed crossing the boundaries of the development team, in inter-organisational networks rather than in a single organisation. This idea is in line with work in the area of new organisational models, which studied the mechanisms organisations put in place to perform boundary-spanning collaboration [13, 36, 9]. We can see the job shifts as having occurred through *collaboration* in the dynamic, complex and tool-empowered environments of outsourcing as predicted by Appelgate's organizational change framework [46]. As stated in Section 5.1, according to Applegate, in a dynamic, complex and uncertain environment, collaboration becomes a critical organisation design criteria. This is the case not only inside one single organisation, but also between organisations in an inter-organisational network. Therefore, this new current networking condition call for an explanatory theory of software process improvement through the lens of inter-organisational systems. Such a theory would provide new models that explain complex, dynamic inter-organisational software process improvement, e.g., by proposing metrics and visualizations that help IT managers to determine which potential collaborations seem most fruitful.

5.3 Lessons for outsourcing managers

The following guidelines for software development in an outsourcing context result from comparing traditional industrial styles of development with cooperative development processes based on the literature. We studied experiences reported in the literature from a practitioner's perspective, focussing on the sets of metrics and principles provided by our framework. Therefore, we view these guidelines as new insights that need to be validated in further work.

Cohesion and coupling Our first guideline is to analyse coupling and cohesion to improve development processes. By cohesion we mean the number of intra-organisational activities that generate knowledge, such as the amount of time spent in introducing new members to project tasks. By coupling we mean work that crosses the boundaries of a working group or organisation. For instance, coupling increases when interactions between working groups increase. Considering the significant amount of tasks related to coordination in distributed software development, we believe that measuring coupling and cohesion has the potential to paint a picture of where resources go in a project.

Distribution of work Our second guideline is to measure how work is distributed within the project (for instance, employee participation). In offshored outsourcing projects there is most likely little if any control on work distribution within part of the team, as distance creates a large obstacle to exercise control in this way. Measuring the distribution of work enables managers to improve control in the development process.

Mechanisms oriented to understand work distribution that, in particular, support a healthy relationship with the customer of the outsourcing project could shed light on improving outsourced software development. Keeping a healthy customer-vendor relationship during development is essential under today's highly competitive outsourcing market. In this market, with work in conditions of high speed, recent research has shown that not only product quality is important: low customer perception of delivery quality may rule out a supplier for the next project [37].

Coordination mechanisms Our third guideline is to consider improving traditional coordination mechanisms such as plans, system-level design and defined processes. We have seen cases of successful development projects where non-traditional mechanisms are used, such as social networking-like notifications of commit information in the development of the Apache server.

Mechanisms that have been successfully used in open-source projects have the potential to help managers save valuable resources. Mockus, Fielding and Herbsleb [47] examined the development process of an open-source application by measuring elements of software development such as developer participation, core team size, code ownership, productivity, defect density, and problem resolution interval for the Apache web server open-source software development project. The study shows that a large network of people (400 code contributors) cooperated to develop software and that most of the code was made by a small group of developers (approximately 15 developers). It was expected that these 15 developers arranged a partition of the code, to prevent making conflicting changes. But measurements proved otherwise: parts of the system requiring changes were worked upon by more than one developer, suggesting thus a healthy contribution coordination mechanism based on mutual trust and respect. We hypothesise that part of project success is due to the well-covered coordination information needs (such as in the Apache open-source project); which was supported by the metrics they used.

6 Discussion: Comparison with related methods

The study presented in this paper is based on research papers that report real experiences with metrics in organisations. A related study has been published in 1999 by Rico [21]. Our framework is different from Rico's in two ways. Firstly, while Rico's aim was to be complete, our aim is to provide a small set of metrics that (i) have proven to be useful in organisations and (ii) focus on cooperative software development, specifically outsourced software development. Secondly, our selection contains a number of metrics that have been identified after the year in which Rico's selection was published.

Like other methods for software measurement such as Goal-Question-Metric [51], Practical Software and Systems Measurement (PSM) [53], ISO/IEC 15939, and like models to develop measurement capabilities such as CMMI for Acquisition [54], our framework helps project managers to link measurements to information needs. Of the various levels at which the process of software development can be viewed, if the topmost level is where project managers think about the methodology, and the lowest level is where they think about the concrete measurement implementation, then the methods listed above are at a higher level than the framework proposed in this paper. Moreover, contrary to the methods mentioned above, our framework has been constructed with the aim to help project managers to improve control in software development. Our framework is, therefore, complementary to these methods as it supports the application of these approaches to the problem of controlling outsourced software development projects. In other words, our framework can be used to operationalize a method like CMMI for Acquisition, by providing selected experience in a nutshell, obtained from applying outsourcing needs to the state-of-the-art of software measurement.

In this sense, our framework is comparable to MIS-PyME [55], a software measurement methodological framework for small and medium enterprises. MIS-PyME and our framework can both be used to facilitate reuse, because they put tried-and-tested metrics in front of managers in a format that is easy to digest. In particular, MIS-PyME helps users to derive guidelines to identify and

control key indicators based on previous projects of the organization. Our framework recognises and supports reuse of learning and experience, and, contrary to MIS-PyME, helps project managers even when there is no available data from previous projects in their organisation (because our metrics are derived from referenced literature).

Moreover, our work extends existing methods by focussing on the inter-organisational-network view. Many metrics and guidelines support this view (for instance, cohesion and coupling), which is in line with the inter-organisational networking needs of outsourced software development projects.

7 Conclusion

Despite an abundance of research literature on metrics for software process improvement, it is still difficult for managers to choose a set of measurements that enables them to control software development, especially for software development in a cooperative setting such as outsourcing. In this paper, we present a framework that provides a set of measurements and a set of principles and guidelines for the design of an information infrastructure. The set of measurements has been distilled from the research literature by selecting metrics that have been used in real projects in real organisations (as reported in the literature), and that we believe are most suitable for cooperative software development. The framework has been validated by a panel of experts, who confirmed our findings. The main conclusion is that the set of measurements has the potential to improve control in outsourcing projects, and that the principles and guidelines are potentially very useful for managers to apply the measurements in real-world projects, particularly in an outsourcing context.

In addition to providing a practical framework based on the current state-of-the-art in SPI research, this paper also reflects on the role of software measurements in SPI in today's context. Increasingly, software development takes place in very dynamic environments, under high time pressure, and often crossing the boundaries of organisations, as is the case in outsourced software development. Our results suggest that metrics that support feedback between operational and strategic levels help organisations to succeed in dealing with this new context of inter-organisational development. We conclude that this reflection calls for new directions in SPI research, in which theories are developed that (i) explain the dynamics of semi-automatized feedback between operational and strategic levels and (ii) study metrics for SPI through the lens of inter-organisational networks.

Acknowledgment Thanks to the experts who participated in the validation of our results. We would also like to thank the anonymous reviewers for their useful comments.

References

[1] Iversen, J., and Ngwenyama, O.: 'Problems in measuring effectiveness in software process improvement: A longitudinal study of organizational change at Danske Data', Int. J. of Inf. Mngt., 2006, 26, (1), pp. 30–43

[2] Carleton, A., Park, R., Goethert, W., Florac, W., Bailey, E., and Pfleeger, S.: 'Software measurement for DoD systems: Recommendations for initial core measures'. Tech. Rep. CMU/SEI-92-TR-019, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1992

[3] Paulish, D., and Carleton, A.: 'Case studies of software process improvement measurement', IEEE Trans. Softw. Eng., 1994, 27, (9), pp. 50–57

[4] Florac, W. A., Park, R. E., and Carleton, A. D.: 'Practical software measurement: Measuring for process management and improvement'. Tech. Rep. CMU/SEI-97-HB-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1997

[5] Iversen, J., and Mathiassen, L.: 'Cultivation and engineering of a software metrics program', Inf. Sys. J., 2003, 13, pp. 1365–2575

[6] Palacio, R., Vizcaíno, A., Morán, A., and González, V.: 'Tool to facilitate appropriate interaction in global software development', IET Softw., 2011, 5, (2), pp. 157 –171

[7] Lacity, M. C., Khan, S. A., and Willcocks, L. P.: 'A review of the IT outsourcing literature: Insights for practice', J. Strat. Inf. Sys., 2009, 18, (3), pp. 130 – 146

[8] Ponisio, L., and Vruggink, P.: 'Effective monitoring and control of outsourced software development projects'. In: Song, W. W., Xu, S., Wan, C., Zhong, Y., Wojtkowski, W., Wojtkowski, G., and Linger, H. (eds.) Information Systems Development (Springer, 2011), pp. 135–147

[9] Ponisio, M. L., and Vruggink, P.: 'Analysing boundary objects to develop results that support business goals'. In: 2008 International Conferences on Computational Intelligence for Modelling, Control and Automation; Intelligent Agents, Web Technologies and Internet Commerce; and Innovation in Software Engineering (IEEE Computer Society, Los Alamitos, CA, USA, 2008), pp. 516–521

[10] Heeks, R., Krishna, S., Nicholsen, B., and Sahay, S.: 'Synching or sinking: global software outsourcing relationships', IEEE Softw., 2001, 18, (2), pp. 54–60

[11] Lacity, M. C., Willcocks, L. P., and Rottman, J. W.: 'Global outsourcing of back office services: lessons, trends, and enduring challenges', Strat. Outsourcing: An Int. J., 2008, 1, (1), pp. 13–34

[12] Boehm, B. W., and Sullivan, K. J.: 'Software economics: A roadmap'. In: International Conference on Software Engineering (ICSE) (2000)

[13] Star, S. L., and Griesemer, J. R.: 'Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39', Soc. Stud. Sc., 1989, 19, (3), pp. 387–420

[14] Aaen, I., Arent, J., Mathiassen, L., and Ngwenyama, O.: 'A conceptual MAP of software process improvement', Scandinavian J. Inf. Sys., 2001, 13, pp. 81–101

[15] Grady, R. B., and Caswell, D. L.: 'Software metrics: Establishing a company-wide program' (Prentice Hall, Englewood Cliffs, NY, 1986)

[16] Rifkin, S., and Cox, C.: 'Mesurement in practice'. Tech. Rep. CMU/SEI-91-TR-016, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1991

[17] Grady, R. B.: 'Successful software process improvement' (Prentice Hall, Saddle River, NH, 1997)

[18] Rosenberg, L. H., Sheppard, S. B., and Butler, S. A.: 'Software process assessment (SPA)'. In: Third International Symposium on Space Mission Operations and Ground Data Systems, Part 2, Greenbelt, MD, USA (1994), pp. 1001–1008

[19] Herbsleb, J., Carleton, A., Rozum, J., Siegel, J., and Zubrow, D.: 'Benefits of CMM-based software process improvement: Initial results'. Tech. Rep. CMU/SEI-94-TR-013, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1994

[20] Dekkers, C. A.: 'The secrets of highly successful measurement programs', Cutter IT J., 1999, 12, (4), pp. 29–35

[21] Rico, D. F.: 'Using cost benefit analyses to develop a pluralistic methodology for selecting from multiple prescriptive software process improvement (spi) strategies' Master's thesis, 1999

[22] Hansen, B., Rose, J., and Tjrnehj, G.: 'Prescription, description, reflection: the shape of the software process improvement field', Int. J. of Inf. Mngt., 2004, 24, (6), pp. 457–472

[23] Fuggetta, A., and Picco, G. P.: 'An annotated bibliography on software process improvent', ACM SIGSOFT Softw. Eng. Notes, 1994, 19, pp. 66–68

[24] Conradi, R., and Fuggetta, A.: 'Improving software process improvement', IEEE Softw., 2002, 19, (4), pp. 92–99

[25] Austin, R., and Paulish, D.: 'A survey of commonly applied methods for software process improvement'. Tech. Rep. CMU/SEI-93-TR-027, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1993

[26] Herbsleb, J., Zubrow, D., Siegel, J., and Rozum, J.: 'Software process improvement: State of the payoff', Am. Progr., 1994, 7, pp. 2–12

[27] Daskalantonakis, M.: 'A practical view of software measurement and implementation experiences within motorola', IEEE Trans. Softw. Eng., 1992, 18, pp. 998–1010

[28] Barnard, J., and Price, A.: 'Managing code inspection information', IEEE Softw., 1994, 11, (2), pp. 59–69

[29] Goldenson, D. R., and Herbsleb, J. D.: 'After the appraisal: a systematic survey of process improvement, its benefits, and factors that influence success'. Tech. Rep. CMU/SEI-95-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1995

[30] Hall, T., and Norman: 'Implementing effective software metrics programs', IEEE Softw., 1997, 14, (2), pp. 55–65

[31] Gopal, A., Krishnan, M., Mukhopadhyay, T., and Goldenson, D. R.: 'Measurement programs in software development: Determinants of success', IEEE Trans. Softw. Eng., 2002, 28, pp. 863–875

[32] Iversen, J. H., and Kautz, K.: 'The challenge of metrics implementation'. In: Svensson, L., Snis, U., Srensen, C., Fgerlind, H., Lindroth, T., Magnusson, M., and stlund, C. (eds.) Proc. IRIS 23. Laboratorium for Interaction Technology (2000)

[33] Kautz, K.: 'Making sense of measurement for small organizations', IEEE Softw., 1999, 16, (2), pp. 14–20

[34] Mirani, R.: 'Procedural coordination and offshored software tasks: Lessons from two case studies', Inf. & Mngt., 2007, 44, (2), pp. 216–230

[35] Oshri, I., Kotlarsky, J., and Willcocks, L. P.: 'Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects', J. Strat. Inf. Sys., 2007, 16, (1), pp. 25–49

[36] Kellogg, K. C., Orlikowski, W. J., and Yates, J.: 'Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations', Org. Science, 2006, 17, (1), pp. 22–44

[37] van Ekris, J.: 'Customer perception of delivery quality: a necessary area for attention for project managers'. In: Pahl, C. (ed.) Proceedings of IASTED International Conference on Software Engineering as part of the 26th IASTED International Multi-Conference on Applied Informatics, Innsbruck, Austria (ACTA Press, 2008), pp. 268–275

[38] Gopal, A., and Gosain, S.: 'The role of organizational controls and boundary spanning in software development outsourcing: Implications for project performance', Inf. Sys. Res., 2010, 21, (4), pp. 960–982

[39] Misra, R. B.: 'Global IT outsourcing: Metrics for success of all parties', J. Inf. Tech. Cases and Appl., 2004, 6, pp. 21–34

[40] Klein, H. K., and Myers, M. D.: 'A set of principles for conducting and evaluating interpretive field studies in information systems', MIS Q., 1999, 23, (1), pp. 67–93

[41] Walsham, G.: 'The emergence of interpretivism in IS research', Inf. Sys. Res., 1995, 6, (4), pp. 376–394

[42] Walsham, G.: 'Doing interpretive research', Eur. J. Inf. Sys., 2006, 15, (3), pp. 320–330

[43] Hevner, A., March, S., Park, J., and Ram, S.: 'Design science in information systems research', MIS Q., 2004, Vol. 28, (1), pp. 75–105

[44] Kitchenham, B., and Charters, S.: 'Guidelines for performing systematic literature reviews in software engineering'. Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007

[45] Kitchenham, B., Pretorius, R., Budgen, D., Pearl Brereton, O., Turner, M., Niazi, M., and Linkman, S.: 'Systematic literature reviews in software engineering – a tertiary study', Inf. and Softw. Tech., 2010, 52, (8), pp. 792–805

[46] Applegate, L. M.: 'Managing in an information age: Transforming the organization for the 1990s'. In: Proceedings of the IFIP WG8.2 Working Conference on Information Technology and New Emergent Forms of Organizations: Transforming Organizations with Information Technology (1994), pp. 15–94

[47] Mockus, A., Fielding, R. T., and Herbsleb, J.: 'A case study of open source software development: the Apache server'. In: Proc. 22nd Int. Conf. on Software Engineering (ICSE 2000), Limerick, Ireland (ACM, New York, NY, USA, 2000), pp. 263–272

[48] McLoughlin, F., and Richardson, I.: 'The Rosetta Stone Methodology – a benefits-driven approach to SPI'. In: Systems, Software and Services Process Improvement, proceedings of the 17th European Conference, EuroSPI 2010, Grenoble, France (2010), pp. 201–212

[49] Kan, S. H.: 'Metrics and models in software quality engineering' (Addison-Wesley, Boston, MA, USA, 1995)

[50] Lauesen, S., and Vinter, O.: 'Preventing requirement defects: An experiment in process improvement', Req. Eng., 2001, 6, (1), pp. 37–50

[51] Basili, V.: 'Software modeling and measurement: The goal/question/metric paradigm'. Tech. Rep. CS-TR-2956, UMIACS-TR-92-96, University of Maryland, 1992

[52] Lacity, M. C., and Willcocks, L. P.: 'An empirical investigation of information technology sourcing practices: Lessons from experience', MIS Q., 1998, 22, (3), pp. 363–408

[53] McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., and Hall, F.: 'Practical software measurement: Objective information for decision makers' (Addison-Wesley, Boston, MA, USA, 2002)

[54] CMMI Product Team: 'CMMI for acquisition, version 1.2', 2007

[55] Diaz-Ley, M., Garcia, F., and Piattini, M.: 'Implementing a software measurement program in small and medium enterprises: a suitable framework', IET Softw., 2008, 2, (5), pp. 417–436