Why Do Programmers Avoid Metrics?

Medha Umarji
Dept. of Information Systems
UMBC
Baltimore, MD, USA
+1 (410) 455 3956
medha1@umbc.edu

Carolyn Seaman
Dept. of Information Systems
UMBC
Baltimore, MD, USA
+1 (410) 455 3937
cseaman@umbc.edu

ABSTRACT

Software process improvement initiatives such as metrics programs have a high failure rate during their assimilation in a software organization. Social and organizational issues are some of the factors affecting the adoption and acceptance of metrics, and these issues have not been discussed in detail in existing metrics literature. We undertook an interview-based study with the purpose of studying factors that influence the buy-in of metrics. We interviewed 12 members of the metrics team of a large multi-national corporation, with a thriving metrics program. We found that there was some resistance to standardization of corporate metrics processes introduced by the metrics team. This resistance centered on the metrics data collection and reporting processes. One cause of resistance was the presence of sub-cultures and native data collection and reporting processes within organizational units that were independent businesses before they were acquired. Some of the pushback manifested itself through begrudging compliance, and avoidance activities like scripting and gaming of metrics. In this paper, we present the perspectives of developers, managers and upper-level management to emphasize that each stakeholder in the metrics initiative has a valid viewpoint that should be taken into account while implementing a metrics program and that each metrics effort is inextricably enmeshed with the organizational context. We provide actionable recommendations to understand the different perspectives and to adapt the metrics effort accordingly.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *process metrics, product metrics.*

General Terms

Management, Measurement, Human Factors, Standardization.

Keywords

Software metrics, unit testing, static analysis, code review, quality, phase containment, in-process metrics.

1. INTRODUCTION

In the past two decades, there has been considerable progress in the technical design and understanding of software quality and software metrics. Several methodologies such as the Goal Question Metric (GQM) paradigm [1], as well as the Capability Maturity Model (CMM/I) framework [2] for process assessment have been defined for improving software processes and products through the use of metrics.

While there is plenty of good advice on how to start a metrics program (e.g., GQM) and what the indicators of success are (e.g., use in decision-making, and improvement in organizational performance [3]) there is little knowledge about how a metrics program can be tailored to an organization's cultural and social context, in order to be successful. The research presented in this paper aims to understand the social, behavioral and cultural obstacles to getting the buy-in of developers and managers.

In this paper we present an investigation of the factors that influence the buy-in and routinization of metrics within the dynamics of a large multi-national corporation. We interviewed 12 people from the metrics team at ABC Corporation (name disguised). These interviews were largely exploratory. We leveraged the complex, multi-national environment at ABC to get a multitude of perspectives through our participants who interact with all strata of management and development. Our primary research question was: *what are the problems surrounding full-fledged acceptance of metrics in ABC's software metrics implementation*?

It was encouraging to learn that all the organizational units saw the usefulness of metrics and were interested in collecting metrics to learn about how they were doing internally. However, other organizational dynamics were at play, such as the fact that they were hesitant to report their metrics to the corporate metrics team, and there was a pushback on the standardization of in-process metrics across all organization units. This hesitation and pushback resulted in practices that we term gaming, i.e. avoiding full compliance with the metrics program while appearing to provide all requested data.

The metrics literature is bereft of case studies that discuss in rich detail the trials and tribulations of metrics data collection and reporting, and our work is an important contribution. Although we do not have sufficient evidence to link these issues directly to the success or failure of a metrics program, several studies have reported that these nuances could impede the progress of a well-designed metrics program [4, 5].

The problems we report, however, are only a small part of the story of metrics at ABC. It is important to note at this point that despite our discussion of the problem of gaming and other negative practices, the metrics program at ABC was actually quite mature and successful. The reasons for, and indicators of, the success of metrics at ABC are many and varied, but are outside the scope of this report, which focuses on one type of difficulty that could arise in any metrics program.

In the following section we discuss the related work on metrics programs. In Section 3, we present a detailed report of the research methodology that we used for this study. Sections 4 and 5 discuss the background of the organization and findings pertaining to gaming of metrics, respectively. We follow on with a set of implications for research and practice in Section 6 and Conclusions in Section 7.

2. RELATED WORK

We discuss below some of the issues that have been raised by prior empirical work in metrics programs. Based on the prior research in this domain, we report representative findings about people issues that can influence the acceptance and use of a metrics program by developers.

Gopal, Mukhopadhyay and Krishnan [3] discuss that usage of metrics in decision making and improved organizational performance are key **indicators of success of metrics initiatives**. It was found that there is a cyclical relationship between the two indicators of success: increased use of metrics in decision-making leads to increased organizational performance and vice versa. Having a systematic process of data collection and analysis in place has a significant influence on use of metrics in decision-making. Similarly, collecting the basic and advanced metrics (basic first, followed by advanced) also has a positive influence on this variable.)

Metrics data being used for **project comparison purposes** is one of the potential causes for dishonesty in reporting metrics, by project managers and developers. Herbsleb and Grinter [5] report that if uniform definitions are called for across projects, it can be perceived as a threat by the employees because it raises the possibility of being compared across projects and organizational units. They reason that this may make developers reluctant to share their data due to apprehensions about unfair comparisons with other projects.

In contrast, an example in which such uniformity was not pursued is that of Contel Corporation, discussed by Pfleeger [6]. At Contel, project managers were given the flexibility of choosing a metrics tool, best suited for their purposes. Pfleeger [6] and her colleagues adopted this approach, primarily because the projects were very different and collecting uniform metrics across projects would have been problematic. Since projects had considerable flexibility in choosing the metrics tools, and types of data and analysis to be performed, managers as well as developers had a **feeling of control** over the metrics program. Pfleeger reports that as the metrics were local to each project and were used for process improvement pertaining mainly to that project, managers as well as developers were genuinely interested in the results of metrics data analysis, and there were no doubts about the usefulness and integrity of the data. This study is the closest to our work, as it closely matches the situation at ABC. Unfortunately, Contel had to shut down during the attempt to integrate metrics across different projects.

Another finding of the Herbsleb and Grinter study was that developers do not believe in the **integrity of metrics** data. Accordingly, they do not whole-heartedly participate in metrics activities. Fenton and Hall [4] conclude that it is very important for practitioners to believe that the data is accurate and reflects reality, i.e. it satisfies the representation condition. Fenton and Hall [4] also found that some developers believe that the data will be massaged (by their managers), and this affects their motivation to collect data, since they feel that it will be modified anyway.

A related issue discussed by Fenton and Hall [4] is that of **differences in thinking between managers and developers**. They found that managers do not believe that developers are receptive about collecting metrics, and this in turn influences the thinking of developers. Weinberg [7] observed that faking time sheets is a part of the universal culture of software development and that "programmers tend to tell the lies that their managers want to hear". Developers' acceptance of metrics activities as part of their work practices is, to some extent, within the managers' control [4].

Another issue is that **developers feel threatened** that metrics might be used against them. Dekkers [8] reports that there should be a safe environment for collecting and reporting data. Once the developers are convinced that metrics data is being used for measuring the process and the products, not them, then their resistance will be reduced. For this purpose, Iversen et al., [9] suggest that developers should have complete access to data they submitted, in order to keep the process transparent. They claim that data collection routines should be tied to the project milestones and treated as deliverables along with other deliverables of the project. They report that impressions about the metrics program are formed through the informal channels of communication and are influenced by past similar initiatives, if any. They find that this influences developers' perceptions of the current initiative and in turn, their acceptance of it.

Most studies conducted on metrics programs have stressed the importance of using **automated tools and techniques** for collecting data [6, 9, 10]. The rationale behind using automated tools is that it reduces developer resistance to the metrics program by reducing the burden of extra work, by helping to ensure the validity and integrity of the data and helping in the presentation and evaluation of metrics data. However, it is not always possible to collect accurate data without active developer involvement [4]. Dekkers [8] recommends realigning the corporate reward system to promote collection of complete and accurate data thus tying a formal incentive structure to the metrics program, and encouraging developer involvement.

Based on these findings from the literature, it is clear that there is awareness about the fact that gaming occurs in software metrics. Many solutions have been proposed to stop gaming and collect accurate data. However, these solutions have limited success because metrics data collection is a practice that is very organization-specific and very people-specific. So recommendations must be tailored to that organization's specific context and the opinions and perceptions of people collecting and reporting those metrics.

3. METHODOLOGY

As described in the Introduction, we were working with ABC Corporation, a large, multi-national, U.S.-based networking technologies company, to understand the complex dynamics at play in their corporate-wide roll-out of metrics. We used a qualitative research methodology, specifically unstructured interviews, because we sought to have an in-depth understanding of organizational, metrics-related and people-related factors. It is well documented that such insights could be obtained through qualitative research techniques such as interviews and observations [11].

Our contact at ABC helped us in identifying and talking to people that would give us an overall view of corporate metrics in an unbiased manner. The sampling strategy was purposive, as we chose each of the participants for a specific reason and their inclusion was intentional. All of the 12 interviewees were chosen based on their particular position in the company such as the group they worked for or the tool they had created or the process they had championed. However, as in any industrial case study, we were constrained by the type of people we had access to. For this phase of the study we did not have access to current developers, but only to people who were actively involved with metrics across multiple projects. The advantage of this sample is that each of our participants had interacted with several hundred developers and managers as well as with higher levels of management. So while we did not have a pure developer perspective, our participants brought a much more rich and diverse set of perspectives to the study.

All the interviews were held on a one-to-one basis, either through telephone or in-person. All the interviews were recorded and transcribed, in addition to taking notes. Of the twelve participants we spoke to, three were champions of three practices i.e. static analysis, unit testing and code review. Two of the respondents were people in charge of the dashboards for product and in-process metrics respectively. Two participants were leads in the Quality Improvement initiative (QII), which is the product quality initiative at ABC introduced 4-5 years earlier than the process initiative. Five of the respondents were on the corporate metrics team that makes the decisions for rolling out programs on a company-wide basis.

Our interviewing strategy was to ask simple questions about the background and components of the metrics program, with the expectation that the participants would give us more information voluntarily, in the course of the conversation. We did not ask all the questions to each participant and other questions were asked, to keep the flow of the conversation going. Since we intended to use a grounded theory approach our interview questions were fairly open-ended and were used more as starting points for a conversation. In cases where a participant was talking about related issues, we did not stop them. Consequently we received a rich, detailed account of each participant's perspective of the current and past metrics programs at ABC Corporation.

The interview prompts included but were not limited to:

1. Could you tell me about the history of metrics in this organization?
2. What were the goals behind introducing the metrics program?
3. In what way was the program introduced, example: by piloting one unit, or an organization-wide adoption?
4. How many and which metrics does this metrics program collect?
5. Is it mandatory for organizational units to participate in the metrics activities?
6. Was there any kind of training given to organizational units, prior to starting the metrics program?
7. Was this metrics program marketed? If yes, how?
8. How far has this metrics initiative met the goals it was designed for?
9. Is there any other information that you think we should know?

The collected data was analyzed using a grounded theory approach. We started with open coding, where passages in the interview data were identified as pertaining to specific themes or topics, then related passages were sorted and grouped. The process continued with axial coding, in which emerging patterns were noted. Findings were documented through selective coding, which resulted in this paper [12].

Admittedly in a case study of this nature, there are several threats to validity of the results [13]. Foremost is the issue of generalizability of findings. Keeping this limitation of our study design in mind, whenever possible, we have compared and contrasted the results of our study with other studies in the metrics literature. In cases where the findings are not supported by the literature, we have been very cautious in drawing conclusions from the data.

Our findings about process-related metrics are limited to those related to the three practices of code reviews, unit testing and static analysis, even though the organization has several other practices in place for ensuring software quality. We were focused on these three practices partly because they were of interest to the organization and partly because these practices are widely deployed in the industry and thus our study results would have a potentially larger impact.

Since our interviewees were primarily the people involved in the corporate metrics team, they were sensitive to the introduction of incomplete or inaccurate data. The interviews and qualitative analysis were performed by one author, and the other author thoroughly reviewed all the findings from the analysis. During the interviews the focus was on the difficulties of getting a metrics program accepted by developers. This may be a source of bias for our study. To mitigate this bias, we shared all our findings and conclusions with our contact at the organization and had them verified for accuracy. Our contact provided useful feedback and contributed additional explanations for our conclusions. The opinions expressed here are based principally on the intuitions and perceptions of people on the metrics team, and should be treated as such.

The other concern - that of objectivity of the data - has been addressed by our sampling strategy as we have taken the precaution of talking to people from different groups within the company, not just one particular team or unit.

4. ORGANIZATIONAL CONTEXT

From its inception, ABC has expanded greatly through acquisitions of several small companies. ABC currently has over 65,000 employees worldwide. Typically each business group or organizational unit corresponds to a start-up company that was acquired by ABC at some point. Organizational units have their own sub-cultures, their own tools and processes for collecting metrics and their own indicators of quality.

There is a focus on using and developing in-house tools, rather than adopting "academic" metrics frameworks such as the CMM/I family [2], and we found that there were many people who were passionate about metrics, including our interviewees.

Overall, ABC is a highly motivated and driven company that is knowledgeable about the good aspects of metrics. Due to their 'silo-ed' structure and history of acquisitions, communication across different units is strained and there is a competitive spirit between different organizational units.

4.1 The quality improvement initiative

In order to understand the metrics culture at ABC, it is important to understand and discuss the hugely successful Quality Improvement Initiative (QII – name disguised) that was introduced many years ago at ABC Corporation.

The QII was introduced to monitor product reliability and to avoid frequent bug-blitzes. Some of the metrics collected for this initiative were mean-time-to-repair (MTTR), customer found defects (CFDs) and internally found defects (IFDs). Due to the fact that customers, products and markets differ across each organizational unit, each unit was allowed the flexibility of developing their own metrics for quality (as long as they included MTTR, CFDs and IFDs), and setting goals around those metrics.

The metrics team that implements QII works with customers and designs the customer satisfaction survey, results of which are factored in to employee bonuses; they also serve the role of a free press – an unbiased voice that conveys the state of quality and customer experience to rest of the company; and thirdly, they work with organizational units to help them improve their quality and resolve any other quality-related problems.

4.2 The process improvement initiative

Two years before the study reported here, the corporate metrics team (a sister concern of the quality metrics team) decided to go a step further than product quality, and focus their efforts on measuring *phase containment* i.e. catching defects during the phases of software development and testing, rather than later, in the field. The underlying assumption was that all the defects that are not contained in development, ultimately end up as customer-found-defects (CFDs).

Phase containment is an umbrella term for a range of activities and metrics at ABC. One of the phase containment related activities is the escape detection process (EDP). It is an elaborate subsystem that assesses defects that flow from one phase into the next downstream phase. EDP involves the collection of numerous metrics. EDP-related metrics, as well as other metrics associated with phase containment, such as defect density and mean-time-to-repair, are considered in-process metrics because they are indicators of an ongoing process, rather than a finished product. In our interviews we focused on metrics related to code review, static analysis and unit testing. These phase containment practices are used widely throughout ABC.

At the start of the initiative to measure and track phase containment, each organizational unit had their specific way of collecting and analyzing in-process metrics. Each organizational unit performed a subset (or superset) of the phase containment activities. There were some groups that were very active in performing unit testing, some in static analysis and others in code reviews. Looking at the success of these practices at the organizational unit level, the corporate metrics team then decided to institutionalize these three practices, and actively urged all the organizational units to adopt them.

The phase containment and overall process improvement initiatives were mainly an effort to standardize these practices and metrics around them, across all the organizational units. As will soon be evident, much of the pushback originated from this effort to standardize practices and metrics – as it raised the possibility that different organizational units would be compared with each other based on their metrics.

Due to the large number of small organizations at ABC and their tendency to have their own tools and practices, it was impossible for the metrics team to design metrics around these three practices that would account for work practices across all the organizational units. Therefore, they decided to create a very simple metric to begin to gauge adoption of the three practices. Each code commit and each bug report had to include attachments indicating whether or not these three practices had been performed on that piece of code. A count of such attachments constituted a rough measure of adoption.

The adoption measures for different practices can be seen through dashboards, with aggregated views for different levels of the organizational hierarchy.

4.3 Dashboards

Dashboards are at the intersection of these initiatives and the metrics data. In this section, we discuss the quality dashboard and the process dashboard, which represent data from the quality improvement and process improvement initiatives respectively.

The quality dashboard provides data on metrics such as mean-time-to-repair (MTTR) and customer found defects (CFDs) to different levels of the organization. The process dashboard is built on top of a database of code commits and defect reports (each with their attachments, as described earlier). The dashboard queries this database for attachment-related information and presents data to the users by drill-down and roll-up operations, based on the user's position in the organizational chart.

Dashboards are the interface of metrics program at ABC. The dashboards at ABC are enormously successful. A program manager is constantly monitoring the dashboards and is always around to answer questions and quell doubts. The program managers of these dashboards strive to make sure that there are no errors in the reported data. The dashboards are thought to be very intuitive and there is no training required for using any of the dashboards.

From our understanding of the dashboards and the two metrics initiatives discussed above, we believe that the corporate metrics team at ABC is doing the best they can in implementing metrics and standardizing them across different organizational units.

5. FINDINGS

In this section we start by presenting some of the general findings from our interviews. We believe that behaviors such as those we've observed are evident in many metrics programs, and so we highlight these findings so that other organizations can learn from them. Based on our interview findings, we also present the various social, psychological and situational impediments to corporate wide adoption of effective metrics data collection and reporting strategies, which in turn can influence the overall health of a metrics program.

5.1 Problems with data collection

In our interviews, some of the problems surfaced surrounding the complete adoption and acceptance of software metrics. Many of our participants revealed interesting insights and stories about how developers were resistant to collecting metrics and, in some cases, how metrics data was manipulated.

Some of the ways in which metrics data was (intentionally or unintentionally) misrepresented are discussed next.

Suppressing partial information: When developers (and managers) wanted to comply with metrics processes, but did not want to report bad information about their colleagues and friends, they would report only the really important bugs and suppress other information. An example is suppressing the reporting of defects with lowest severity.

One of our participants, who was actively engaged in training developers from several different organizational units to perform effective code reviews, had this to say about reporting code review related metrics such as bugs found:

*"So one of the problems that we have… in China is that… do not make your friends lose face. So they only write up the high priority and medium priority defects. They don't write up low priority defects. Well… usually it's the low priority defects that *clarify* something going on in the comments… Maybe one defect by itself isn't a big deal but when you put them all together it makes something really difficult to understand."*

This class of metrics reporting problems seems innocuous and is probably the least harmful, but the prevalence of such behavior highlights the fact that programmers are very sensitive about what information is released about them into the organization – especially defects found in their code are taken very personally. These subtleties should be taken into account by managers and peer developers when performing processes like code reviews.

Scripting: This is the act of writing scripts to enter values within specific data ranges into metrics reports. The idea is to demonstrate that metrics are being collected, while at the same time avoiding the unpleasant task of reporting them manually.

The adoption metric used at ABC was a target of some scripting. Recall that the adoption metric was a measure of whether a bug or work item had attachments pertaining to code review, unit testing and static analysis. The contents of the attachments could not be verified completely in terms of accuracy of reported data. Usually, only the presence or absence of an attachment was logged regularly by the metrics team. Therefore, in some cases teams would resort to scripting (i.e. writing a script to automatically generate an empty attachment) in order to push their adoption numbers up.

One of our participants was in charge of monitoring the dashboards that reflected how often phase containment practices were performed within each organizational unit, and in the participant's words:

"...you can just put your adoption numbers up if you are doing a unit test, by putting a script that will generate the attachments. And we call that gaming the metric."

Scripting is also one of the obvious side-effects of begrudging compliance. As our participant who is an expert on unit tests puts it:

"And you can put a statement on that, you can say, "Yes. I performed unit tests and there were no problems." So, sometimes, when we look at the bug data, we see attachments that were added by scripting, you know, at night."

Entering false data: Scripting also enters false data, but in some cases metrics are intentionally falsified while being entered into metrics reports in order to appear compliant and to make numbers look good. This class of misrepresentation is a cause for worry in any metrics implementation.

It is difficult to detect if false metrics data has been reported. Sometimes a member of the metrics team might find a team's numbers suspicious like one of our participants did:

"It's just that their numbers are kind of suspicious, because they are reaching the upper average of 80. Earlier we were average of 70, now we are average of 80, before that we were supposed to reach 95 percent, so lot of people were aware of that, so lot of people you know were trying to jack up their numbers…"

Falsifying metrics data is a very common problem with metrics programs and if it happens on a small scale, it goes unnoticed. However, it is important to realize that this class of problems with metrics reporting is not only a political issue it is also an issue of priority. The fact that people resort to gaming could be interpreted to mean that metrics are not perceived as useful and meaningful, rather they are viewed as an afterthought and taken lightly.

Illusion of compliance: In some cases, people just give the illusion of participating. Like one participant said -

"Sometimes when we look at attachments, they will be blank, there will be no data… that just makes me mad"

In this case, the team or organizational unit does not want to misrepresent information – but because of the way that the adoption metric works, they may still get credit for submitting an empty attachment.

Another participant had something similar to say about metrics data:

"So when they [dashboards] say 95% have adopted this metric, they [dashboards] mean 70% because a chunk of them [attachments] are [useless]."

The problem of illusion of compliance is really a reflection of loopholes in the design of metrics collection and reporting processes. It is relatively easy to circumvent this problem if the right controls are added in the metrics processes, for example – a mechanism to detect empty attachments in this case.

There are many suggestions in prior literature (as discussed in Related Work) to circumvent these problems in data collection and reporting. For example, a preference for automated data collection has been emphasized, especially as it increases the quality of metrics data. Automation might, in some cases, make various forms of gaming more difficult. On the other hand, removal of a human to check the data being submitted might actually facilitate other forms of gaming, and as is evident from these results, especially in organizations with such a broad range of metrics and different metrics being applicable to different projects, automation has a dangerous side as well. For example, when the overhead of reporting metrics manually is too high, people perform the required practices (like unit tests) but write scripts to fill in the reports.

5.2 Reasons for pushback

In the previous section we discussed different classes of problems in metrics data collection processes. In this section we focus on some of the reasons that these problems arise. It is worthwhile to note that even though these problems are specific to ABC, they can be observed in different forms across organizations, as documented in several prior studies of metrics programs. In this section, we present views about management as well as developers.

5.2.1 *Maintaining image*

At ABC, people were concerned about their image. Moreover in some cases, the polarity of their feelings towards metrics was based on whether metrics data enhanced their image.

One of our participants worked extensively on the product quality dashboards and based on his experiences with publishing the data, he told us that:

"People feel like you are measuring them, well if you are measuring them, and they are doing good, it is OK."

This sentiment was echoed by two other participants:

"Of course as soon as we provide the information, some people say, "Oh, this is good!" because their numbers look good, some will say - this is not right, because you know ..."

"(People say) but before you put it on the dashboard, can you show me how I'm going to look? I don't want to be embarrassed."

Therefore we glean that managers and developers do not mind collecting and reporting metrics as long as it does not cause them to look bad in front of their superiors and colleagues. The converse is also true, i.e. in cases where metrics data reflects poorly on a unit, they will oppose the publishing of that data, and find fault with the metrics program, e.g. protesting that the metric does not adequately represent their processes.

However, it is important to realize *why* managers and developers are willing to fight tooth and nail to make sure that they maintain a good image.

The first reason is that people genuinely believe that they are doing good work. Through the interviews, we learned that developers and managers alike were extremely passionate about their projects and they took great pride in their work. Therefore it was unacceptable for them to see metrics data inadequately representing the amount of effort they had put into the process.

This quote from one participant who was the program manager for a dashboard reveals how passionate people are about their work:

" (the process) dashboard is pretty accurate, because people fight with you if you are wrong for one bug…, they can easily see how many bugs they have, if it is different from their list they are going to fight with you, so we have to make it very clear and very accurate. If you publish the wrong data, they are going to fight with you, they are going to escalate… and you know your dashboard is gone, nobody can use it."

A related issue was that developers take pride in their code, and they like to do things their way. Having someone else find bugs in their code, for example, hurts their pride. One of our participants who had close to thirty years of experience in working on metrics pointed out that:

"If (a programmer's) code has more bugs than somebody else's code, they don't want anyone to know about it. So next time, they may not record the number of bugs found, or they will not do code review until they are pretty darn sure that their code is good."

We also learnt that in some instances there was a disconnect between actually performing a phase containment activity and reporting metrics about that activity. Our participant who specialized in code reviews reported that:

"One of the problems we are finding is that people actually do the reviews, they collect the measures but they don't report them. Well the reason for that is the cultural issue. People don't want to find defects in their friends' code."

In such situations, one cannot help but sympathize with the developers or managers. Software development is a very social activity and if metrics were allowed to strain social relationships, they would in fact have a negative impact on collaboration and this could potentially impact the product quality.

5.2.2 *Resistance to standardization*

As discussed in Section 4, each organizational unit was a small company by itself before it was acquired by ABC. All the companies had some form of measurement and improvement processes that they had tried and tested, and that had worked for them. Therefore there was a great deal of resistance to changing from the way things were done while the unit was an independent entity.

One of the participants interfaced with product groups and had to convince them to adopt the corporate metrics and processes. Some of the resistance that he faced was:

"…because you (metrics team) want to change, why do we have to change this process, if we change this process, we will have to change all the big processes associated with it, you know its going to be a big headache, and its not going to be effective"

"…we are publishing data every week, sometimes every day, listen we are not doing static analysis every day, we are doing that once a week, some organizations twice a week, because we want to get together and do it all at once, if something comes up then we will fix all at once, so that's our schedule."

As can be seen, the organizational units in some cases had legitimate reasons for not wanting to participate in the standardization of metrics initiated by the corporate metrics team.

Another common theme was that developers considered their projects to be unique, and did not believe that data about their projects would lend any insight to the metrics implementation.

"... It is impossible to standardize anything here because everyone thinks their project is special..."

It is also interesting to note that the resistance was not against collecting and reporting metrics data, but against the standardization process. When interpreted with our other findings, one possible explanation (in addition to resistance to change) could be that standardizing processes and metrics would immediately lead the metrics team to compare between different organizational units and this was not acceptable to those units. Another barrier to standardization of these practices is that the adoption metric gives rise to data that does not represent effort equally, and results in an unfair depiction of work.

5.2.3 *Begrudging compliance*

By begrudging compliance we mean that developers and/or managers have not bought into metrics, but they are forced by their superiors or other influential people or circumstances to collect certain data. So there is a dogged determination in collecting these metrics but the focus is only on following the process and not on producing useful metrics data.

As one of our participants put it, the attitude is:

"You asked me to do this, now I am going to show you what a bad idea it was."

Sometimes, people are determined to report the metrics they were asked to, but do not show an active interest in how the data will be used, how it will provide a feedback on their processes and whether they should be reporting some other metrics in conjunction with the current ones to provide a complete picture of their work.

"If you tell them you are measuring them on these three things, all of a sudden their world shrinks to those three things. And nothing else matters! And even if they know that not focusing on these two other things is going to hurt overall quality, they will say - hey my manager told me to focus on these three things…"

While reluctant compliance may be better than no compliance at all, it is just as tricky especially because the quality of metrics data may be affected and not much can be learned from metrics data of poor quality. Therefore, while begrudging or reluctant compliance may be a good start for a metrics effort, care should be taken while interpreting data obtained from these sources.

5.2.4 *Metrics are not representative of effort*

It is disheartening for developers and managers to see efforts go unrecognized in the metrics data. Especially if a metric is not designed to distinguish between amounts of effort put in by different groups, there is a lot of resistance and people lose interest in the metrics program.

"*One bug counts as one item in the dashboard for 10,000 lines of code. And then somebody makes a one line code change on a bug fix and that counts equally…*"

Another example is that the metrics that are typically used for code review are not representative of the code quality and the reviewer effectiveness. Two of our participants discussed this issue:

"It is difficult to put a goal on finding defects in a code review - More is better or more is worse? Well, more is good because the person doing the review did a good job, but it is bad because the coder did a bad job…"

"If Code Review A found more defects than Code Review B, it was probably because they were looking at buggier code… and more defects are still there in the code…" [Clarification – In this quote the participant refers to the fact that more bugs does not necessarily mean that the programmer had created those bugs, it could also mean that he was working on a buggier piece of code to begin with].

5.2.5 *Product and process related metrics*

At ABC Corporation, we found that product-related metrics were a part of the routine work practices – measures of customer experience were even tied to incentives. Product quality measures such as CFDs are reported by customers, or are measured by a post-hoc analysis of the product. Therefore the data is objectively verifiable and difficult to misrepresent. A strong focus on product metrics may explain why the Quality Improvement Initiative was such a success.

The perceptions were different, however when it came to implementing in-process metrics and the process improvement initiative. For starters, in-process metrics are much easier to misrepresent as discussed by one of our participants:

"….It is very hard for a sales person to game how many sales they make. But it could be easy to game the in-process metrics they use. It does not necessarily result in sales of the product but it makes their numbers look good."

When it came to in-process metrics, there was a lot of resistance to routinization, and several attempts to achieve uniform adoption of tools and metrics for code reviews, unit testing and static analysis were unsuccessful. Most of the metrics related to these practices could be objectively and automatically assessed. However, since they reflected developers' daily work, developers were not comfortable with sharing that information. Prior attempts at automating such metrics collection had been met with resistance, and as discussed previously the company had to rely on self-reporting.

Some of the resistance around in-process metrics stemmed from a suspicion about the role of the corporate metrics team (for process metrics) and how the data would be used. There seemed to be a general feeling that it was fine for the corporate metrics team to monitor product quality, but when it came to individual process improvement practices, the project or team manager should be given the authority to deal with metrics data, as they are closer to it and understand it better.

On the other hand, one respondent mentioned that it was unfair to evaluate people and organizational units based on *"end-results metrics"* (i.e. product metrics), such as customer experience. One reason was that these metrics were not actionable. Developers did not know what they could change, in order to impact the product quality metrics, as they were calculated at the end of the product development lifecycle.

Thus we conclude that although product-related metrics are easier to measure they are not actionable; and while in-process metrics are hard to measure they provide immediate feedback for process improvement. For an effective metrics program, it is necessary to have both.

6. RECOMMENDATIONS FOR PRACTICE

In this section, we synthesize our understanding of the metrics-related difficulties we found in our study by describing two strategies for discovering and addressing the attitudes and perceptions that can lead to such problems. We first describe contextual interviews as a mechanism for tailoring the metrics program. Then we discuss our prior work in metrics acceptance. Our suggestions are based on preemptively addressing problems that can arise in a metrics implementation.

These strategies can be employed by an organization during the planning and early implementation phases of a metrics initiative, so that mitigating actions can be taken early on if there is evidence that problems may arise. Contextual interviews and the survey to gauge metrics acceptance (as will be discussed next) should ideally be used together. Interviews should be conducted before the start of the metrics program and the survey should be conducted within the first three months. The reason is that the survey assumes that developers have an idea of the impact that metrics will have on their routine work activities. These two techniques assume that there is no problem with the actual design of the metrics program and that management has valid, well-defined reasons for implementing metrics.

6.1 Contextual interviews to facilitate metrics

Contextual interviews are an excellent way of learning about the organizational background, attitudes, opinions and overall perceptions[11]. Talking to the right people can give a well-rounded picture of metrics, and can also give insights into problems specific to that organization, which were not foreseen by metrics implementers.

As is documented in several case studies of metrics, poor quality of metrics data can be a threat to the overall health of the metrics program. Our data revealed that gathering different perspectives in the organization gives a good indication of whether gaming occurs, and also how it can be stopped. Factors such as those that influence gaming are very specific to an organization's context and design of metrics, and it would be fruitless to explore blanket techniques to avoid gaming. Therefore we discuss ways in which information can be used to tailor metrics implementations.

In our study, contextual interviews enabled us to get an in-depth insight into the ways in which metrics data was being reported by the different organizational units. As well, we learnt that the presence of "sub-cultures" was a major reason for resistance towards metrics. The organization units already had a "metrics mindset" and were collecting their own metrics, but they did not want to be part of the corporate program. One respondent even pointed us in the direction of a solution – by recommending that in-process metrics should be kept private to each project group, and product related metrics should be reported to the corporate metrics team.

Such organization-specific issues are not incorporated in any textbook. They can only be learnt by studying the given context thoroughly, from different perspectives of people within the organization. However, it is important to note that since we were a neutral third party and had no influence on any decision-making people were very candid and open with us. We are concerned that it would not be the case if someone from the same organization were to conduct these interviews.

Therefore it is indeed an open, and very sensitive, issue who should conduct these interviews. Our approach was to serve as a neutral third party, and we feel there are some advantages to this approach. We were able to get a "feel" for such issues, and identify potential roadblocks, without preconceived and personal perceptions based on past experience. On the other hand, an "insider" might be better equipped to interpret the findings in light of the organization culture, and to make recommendations that are better tailored to the company. In either case, the management should also start avenues of open communication with developers and managers and design brainstorming sessions around specific practices, such as code reviews, that would identify potential usage problems. The feasibility of doing such an investigation is of course a concern, but given the trade-offs, and based on the rich data that we obtained through our interviews, we think it would be worthwhile in most cases.

6.2 Metrics Acceptance Model

In the context of this study, our prior work on metrics acceptance deserves a mention. We created and validated a list of predictive factors that captured metrics context, opinions and attitudes toward metrics, and give an indication of a person's intention to whole-heartedly participate in the metrics program. Our model, known as the Metrics Acceptance Model (MAM) [14], is grounded in the literature on metrics programs and social psychology, and on our own experience with metrics. The MAM is currently undergoing empirical validation and evaluation, and preliminary results are very promising. It is one of the ways of diagnosing whether developers have bought into the idea of collecting and reporting of metrics.

In practice, we recommend the use of the MAM and its associated questionnaire, to identify potential problems (such as those we found at ABC) before or during early implementation of a metrics program. For more details on the MAM refer to [14].

The MAM is made up of several constructs and each construct tackles a specific problem in metrics implementations. One of the constructs in the MAM is about **social influences**. If one developer feels that the metrics program is not effective or worthwhile, this belief may spread through the group. Or if the manager is not keen on metrics, developers may not be very interested in participating in the metrics program either.

One of the main problems with software metrics programs is that costs are immediate and rewards are long-term. Developers have to have an understanding that metrics are immediately useful to the organization even if their benefits to developers are not immediately evident. **Organizational usefulness** is another construct in the MAM and it checks whether developers have an understanding of the organizational usefulness of metrics. It also inquires about whether developers think that metrics about their project would be useful to the organization.

In an organizational setting image and visibility are very important to a person's career, and this was evident at ABC as well. In fact, "an individual may perceive that using a system will lead to improvements in his or her job performance indirectly due to image enhancement" [15]. Also, people do not like to report problems about themselves or their colleagues as this might affect relationships with them. We found several instances of this reluctance at ABC. The fear of adverse consequences is due to the thought that metrics data can be used to harm a person or someone in his/her social network [6]. These and similar factors are considered to have a positive (or negative) impact on the notion of **personal usefulness** of metrics, and are included in the MAM for that reason.

Factors such as relevant documentation, availability of a help facility, reliable and experienced personnel, adequate time, financial stability, and sufficient documentation are important resources in a metrics program and lack of availability of these can also impact a developers' perception of metrics. For example, if sufficient schedule time is not factored into metrics collection processes, a developer is bound to get harried and have negative perceptions. These factors make up the feeling of **control** that a developer has over the external resources of a metrics program.

Self-efficacy is the belief in one's capability to perform a certain task, and it accounts for internal behavioral control. For example, if a developer is confident that he can analyze metrics data well, that is bound to increase his/her intention to participate in the metrics program.

Compatibility of the metrics tool with existing systems and compatibility of the changes induced by the metrics program with the existing work practices also influence intentions to perform metrics activities. This appears to underlie the resistance of different organizational units at ABC to changing their work processes to accommodate corporate-wide metrics.

Ease of use of the metrics tool and dashboards as well as generally easy to use metrics processes are also vital to a metrics effort. A metrics tool should have an intuitive interface that is clear to interact with and easy to learn. Most prior studies have insisted that the use of automated data collection [4, 6] is very important as it makes metrics easy to use and less cumbersome.

Attitude "is the overall evaluation of desirability of performing the behavior, by the individual" [15]. Attitude has two subcomponents: affective (e.g., happy-sad) and cognitive (e.g., beneficial-harmful). It has been discussed in previous metrics literature that attitude towards metrics could have an influence on the success of the metrics initiative [4].

The list of factors mentioned above is by no means exhaustive, but gives an indication of the types of information that can be revealed through the MAM. Additionally, each construct in the MAM is operationalized through a questionnaire item, and the entire model is essentially a survey of developers' perceptions. Results of this survey can point out areas where developers are dissatisfied with metrics.

It is easily evident that the factors that surfaced at ABC are only a small subset of the factors that are outlined in the MAM. And the MAM is a basically a combination of all the factors that have been encountered in the literature on metrics programs. This just goes to show that problems in metrics programs are highly specific to the organizational context and differ widely across organizations.

7. IMPLICATIONS FOR RESEARCH

We now frame some research questions that can guide the study design and variables studied in future metrics implementation research.

We propose and find support for our assertion that product-related metrics are assimilated more easily compared to process-related metrics. Therefore, the next obvious question is whether it would be better to start a metrics initiative with product-related metrics, as they are less intrusive and more objective as compared to process-related metrics. Such a strategy, if proven to be effective would indeed set the stage for a measurement mindset in the organization and provide valuable indicators to potential obstacles. ABC provides a good example of such a strategy. Although there is some resistance to process metrics, one could easily envision that the resistance would be much higher if there had not been a basic level of acceptance of product metrics first.

Our interview data revealed that "who is asking for the metrics data" has a great impact on its quality and accuracy. If process-related data is to be shared with higher level management, there is a fear that there will be misinterpretation and trouble. Project managers are closer to the developers and are more likely to receive accurate data (especially about the process). However a related finding is that managers are frequently skeptical about metrics themselves.

"We had a survey few years back, and developers were asked basically - does your manager care about quality as much as you do? The answer was no. So when managers tell me that people do not care about the quality, I will just say, well, it is not what the survey shows. The survey shows you do not care."

Hall and Fenton also report a similar finding [4], and therefore it is indeed an open question as to how managers should be first bought into the ideal of collecting metrics. Also, it is interesting to study whether managers should be given autonomy over in-process metrics data as we discussed in the beginning of this section.

An interesting question that emerged from the discussion of national culture is - Does the culture of a country have an impact on metrics implementations? Hofstede [16] has characterized organizations in different countries and they report that in most instances, the culture of the country supersedes the organizational culture. This finding in fact corroborates our conclusion that several key aspects of each metrics program are different, as they are determined by organizational culture, which is in turn influenced by national culture. As an increasing number of organizations outsource their quality assurance activities, and even software development activities to other countries it would be very interesting to study the cross-cultural differences in metrics program implementations.

Herbsleb and Grinter [5] discuss the importance of boundary-spanning roles in large metrics implementations. We also found that, at ABC, the reason that static analysis, code review and unit testing practices got so much visibility and were adopted by the corporate metrics team was that they were championed by people who spanned different organizational units, and sparked interest in different parts of the organization as they moved around. The corporate metrics team at ABC is, in effect, an attempt to span the boundaries of the organization by implementing common metrics. However, it is this attempt at commonality that is encountering considerable resistance. The issue of boundary spanning in metrics program implementation is a complicated issue and deserves further study.

8. CONCLUSION

During the course of our investigation we learnt that if people have not bought into the metrics program, they can resort to unhelpful practices such as gaming and scripting to produce metrics data. If metrics are forced on them, they might go into a begrudging compliance mode i.e. they are not convinced that metrics are useful, and they will prove what a bad idea it really is. We also learnt that each process improvement practice in a metrics program has several nuances surrounding it that may be responsible for the accuracy and quality of metrics data relating to that practice. One of our findings is that product-related metrics are easier to implement and manage than in-process metrics. This may be because process-related metrics can be complex to implement, but easy to game.

From a practical standpoint we propose that contextual interviews by are an effective mechanism for uncovering and understanding the nuances in any metrics effort. Through this technique, an organization can get valuable insights into potential social and cultural problems with metrics. Further, we recommend using the MAM, in concert with contextual interviews, to identify potential problem areas.

From the discussion in this paper, it is easily evident that although overarching issues might be similar, each metrics implementation is significantly different than the other, and the factors vary by organizational context.

We conclude that any metrics implementation is inextricably enmeshed with the organizational context. And the context is determined not just by quantifiable variables such as size, and level of process maturity, but by cultural aspects too, such as the national culture, presence of sub-cultures within an organization, perception that management is intrusive and competition between different organizational units. In order to succeed, metrics programs have to be tailored, which requires that the factors that make each organization unique must be identified. We provide actionable recommendations for doing so.

9. ACKNOWLEDGMENTS

Our thanks to all our participants who took the time to share their experiences with us. Thanks also to ABC Corporation and our primary contact there.

10. REFERENCES

[1] V. R. Basili and H. D. Rombach, *The tame project : Towards improvement-oriented software environments*. College Park, Md.: University of Maryland, 1988.

[2] M. C. Paulk, *Capability maturity model for software*. Pittsburgh, Pa.: Carnegie Mellon University, Software Engineering Institute, 1991.

[3] A. Gopal, M. S. Krishnan, T. Mukhopadhyay, and D. R. Goldenson, "Measurement programs in software development: Determinants of success," *IEEE Trans. Softw. Eng.*, vol. 28, pp. 863-875, 2002.

[4] T. Hall and N. Fenton, "Implementing effective software metrics programs," *IEEE Softw.*, vol. 14, pp. 55-65, 1997.

[5] J. D. Herbsleb and R. E. Grinter, "Conceptual simplicity meets organizational complexity: Case study of a corporate metrics program," in *Proceedings of the 20th international conference on Software engineering*. Kyoto, Japan: IEEE Computer Society, 1998.

[6] S. L. Pfleeger, "Lessons learned in building a corporate metrics program," *IEEE Softw.*, vol. 10, pp. 67-74, 1993.

[7] G. M. Weinberg, *Quality software management*, vol. 4. New York: Dorset House Publishing, 1997.

[8] C. A. Dekkers, "The secrets of highly successful measurement programs," *Cutter IT Journal*, vol. 12, pp. 29-35, 1999.

[9] J. Iversen and L. Mathiassen, "Lessons from implementing a software metrics program," in *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 7 - Volume 7*: IEEE Computer Society, 2000.

[10] D. J. Paulish and A. D. Carleton, "Case studies of software-process-improvement measurement," *Computer*, vol. 27, pp. 50-57, 1994.

[11] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Trans. Softw. Eng.*, vol. 25, pp. 557-572, 1999.

[12] A. L. Strauss and J. M. Corbin, *Basics of qualitative research : Techniques and procedures for developing grounded theory*. Thousand Oaks: Sage Publications, 1998.

[13] R. K. Yin, *Case study research : Design and methods*. Thousand Oaks: Sage Publications, 1994.

[14] M. Umarji and H. Emurian, "Acceptance issues in metrics program implementation," in *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS'05) - Volume 00*: IEEE Computer Society, 2005.

[15] V. Venkatesh and F. D. Davis, "A theoretical extension of the technology acceptance model: Four longitudinal field studies," *Manage. Sci.*, vol. 46, pp. 186-204, 2000.

[16] G. H. Hofstede, *Culture's consequences: Comparing values, behaviors, institutions, and organizations across nations*. Thousand Oaks, Calif.: Sage Publications, 2001.