Implementing Software Metrics at a Telecommunications Company – A Case Study

David I. Heimann
University of Massachusetts Boston, USA

EXECUTIVE SUMMARY

Establishing and using fundamental measures of software development progress is an essential part of being able to predictably produce high-quality customer-satisfying software within a reasonable agreed-to timeframe. However, in many organizations such measurement is done incompletely or not at all. While various forces encourage measurement programs, others form barriers to such implementation.

This case study explores a formal metrics program established to track and analyze the development of a new version for the major voicemail product of a mid-sized telecommunications company. The study addresses the evolution of the company’s organizational structure and culture that led to the adoption of the program, the components and structure of the program, the implementation of the program, its effects on product quality and timeliness, and what happened thereafter. The study also raises questions with respect to facilitating an organizational atmosphere where a metrics program can flourish.

BACKGROUND

This case study investigates a formal metrics program established to track and analyze the development of the major voicemail product for a mid-sized telecommunications-related company, which we shall call “Telogics.” It is described from the point of view of the Metrics Lead for the program.

The study describes how, in an overall downward trend for process and quality assurance, a window of opportunity can open to carry out significant product-enhancing work in these areas. It also raises the issue of how to preserve the gains and historical memory when the window closes and the downward trend guarantees that it will not reopen in the near future.

Aside from the use of metrics to improve the development of a major software product, three other major themes appear in this case:

1. The slow but steady absorption after the merger of two companies of one legacy company by the other one, and the organization evolution brought about but this absorption.
2. The shift by Telogics away from a mature product with a large but static customer base to a multiple line of older and newer products with many new customers, and prospective markets, with an accompanying refocus of Telogics from engineering-driven to market-driven. This maturation of product line with an accompanying flattening of revenues and shift toward new products is occurring within much of the telecommunications industry, not just Telogics.
3. The implementation of a major quality process in a major division of Telogics as the company itself was moving away from a process/quality orientation due to its changing products and organization, and the economic adversity being visited on it and many other companies in the telecommunications industry.

Telogics makes communications systems and software that phone companies use to offer call answering, voice or fax mail, communications surveillance and recording, and other services. Currently (from financial reports in early 2003 for the year 2002), it has around $750 million in revenue, with around $180 million in operating losses and about 4,800 employees. During the period of the case study (approximately the year 2000), it had approximately $1.2 billion in revenue, with around $250 million in operating income and about 6,000 employees. It started in the 1980s as a company producing a large-scale voicemail system.

During the case study period its products were based on two versions of this voicemail system, and they were beginning to branch out into new products such as text-to-speech and intelligent-network applications. It is currently continuing this expansion of its product range, as well as revamping its voicemail-based product after since having eliminated one of the two original versions. As with many telecommunication-based companies, its profits and stock price, very high during the case study period, began declining near the end of that period and has since significantly declined. However, the price is currently stable at the lower level. The corporate organizational structure has evolved over Telogics’ history. Because of the key impact this had on the existence, form, and progress of the metrics program, the organizational evolution during the 1998-2001 timeframe is tracked in detail in this case study.

History

For most of the time in its earlier years, the style of the company (we shall call the company in its earlier years “Techanswering”) was the traditional “cowboy” or “death march” one, very much at the initial process level in the Capability Maturity Model (Humphrey, 1989). There was hardly any process, a lot of what was there were in people’s
heads, their notebooks, or their e-mails. Schedules were determined haphazardly, as were requirements. Towards the ends of projects, Herculean efforts were expended to get the job done, with 70- to 80-hour weeks common. However, frequently the product was delivered late, often with significant major defects. However, with large-scale voicemail being early in its product maturity stage and with few competitors, customers were willing to accept this in order to obtain the then-unique product.

However, it gradually became apparent to Techanswering that the climate was beginning to change, with competitors appearing and customers becoming less tolerant. A major wakeup call was the rejection by a large foreign telephone company of a major Techanswering proposal, a significant setback. Techanswering had had “quality assurance” groups that were actually testing groups for each of the various development centers within the Research and Development organization; but partly in response to this rejection, it formed a Corporate Quality group to guide the software development process for the entire Research and Development (R&D) organization. One of Corporate Quality’s functions was to identify, gather, and report metrics relating to defects and timeliness.

In mid-1998, Techanswering merged with a foreign voicemail system producer, which we shall call “Techvox,” to form Telogics. The two companies had complementary voicemail products and markets. Techanswering built a larger, wireline-based, more complex product, which they sold to comparatively few but large customers, mostly in the United States, South America, and Asia. Techvox built a smaller product, mostly wireless based, which they sold to a larger number of smaller customers, mostly in Europe and the Middle East. As a single company the merged entities therefore commanded a significant market share of a wide-ranging market.

Though the merger was more or less one of equals, Techvox was more “equal” than was Techanswering, an important factor later on during the metrics program. Techvox was financially more conservative, less of a risk taker, and more business-oriented than technology-oriented. Especially initially, Techvox supplied a counterfoil to Techanswering’s let’s-go-for-it somewhat disorganized style, and there are some within the legacy of that company who have said that if the merger had not occurred, Techanswering would have become bankrupt or bought out, considering the way they were doing business.

Despite the more careful style of Techvox compared to Techanswering, it was the latter who continued its interest in software process improvement, quality assurance, and metrics. In fact, by the time of the merger, the scope of its Corporate Quality group had been expanded to cover the whole of the company.

In the spring of 1999, the U.S. product organization (hereafter called The Division) of Telogics was renamed after its primary voicemail product to reflect that its focus was on designing, developing, serving, and supporting this product. Sales and marketing were a corporate-wide function with geographically based divisions, with The Division “contracting” with them to produce orders that these marketing divisions secured. The Division was therefore primarily engineering driven.

**Organization Just Prior to the Case Study Period**

At this point, The Division was carrying out the quality assurance function as follows (see Figure 1): Each of the development centers had a testing group, which
reported to the manager of that development center. They in turn reported to the Deputy Vice President (DVP) of R&D. Within Telogics’ Corporate Quality organization was the Software Quality (SQ) group. The SQ group was responsible for final product quality. It reported to Corporate Quality, a peer of R&D, thus providing an oversight of product quality independent of the development center managers.

Corporate Quality had another key function as well; the Metrics function. This function was to identify, gather, and report metrics for project and corporate-wide activities. This covered metrics for product-related defect occurrences, process-related defect occurrences, and on-time performance.

**SETTING THE STAGE**

**Motivation for the Metrics Program**

Early in the year 2000, The Division delivered a new version of its flagship voicemail product, Version B, to its major customers. Unfortunately, the delivery did not go well. There were a large number of customer-visible defects in the delivered product, many of them major. In addition, frequently the installation at customer sites had problems, some of which affected service. Furthermore, the delivery was several weeks late. Some of the customers warned The Division that this kind of delivery could not happen again if they were to remain customers.
As planning began for the next version, Version C, due out in a little over a year, The Division’s R&D organization, as well as the Corporate Quality group, began to develop a plan to make the quality and timeliness of Version C much more predictable than had been the case for Version B. A significant part of that effort was to develop and carry out a metrics gathering and analysis program (see also McAndrews, 1993; Giles, 1996). This was motivated by two major factors:

1. When a technology reaches the “early majority” or later stage in its maturation, customers regard quality as a major decision factor, whereas earlier in the cycle, new technology features and strong performance are more the major factors (Rogers, 1995; Moore, 1995; Norman, 1998). Telogics’ voicemail product was moving into the “late majority” stage, and the major customers, as demonstrated by their reaction to the mediocre quality of Version B, demanded quality in the product.

2. Defects detected and repaired early in the software development life cycle (SDLC) cost significantly less than if they are detected and repaired later in the cycle. Several sources (Boehm & Papaccio, 1988; Willis, 1998; McConnell, 2001) have propagated a rule of thumb that the cost of a defect increases by an order of magnitude for each step in the SDLC that passes by before it is repaired.

A Major Organizational Change

In early July 2000, just as the metrics program was being planned and begun, a major reorganization occurred (see Figure 2) that would eventually significantly affect the program. Instead of one U.S. and one foreign corporate organization, with the U.S. R&D and marketing departments all reporting to the U.S. Operations Vice-President, each product line became its own division, with its own R&D department reporting to a marketing-oriented VP. Each division would now have its own testing group, reporting to that division’s R&D department. Whether or not a division had a quality group was

Figure 2. Telogics Organization during Case-Study Period

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
left up to that division. The overall Corporate Quality organization became much reduced, concentrating only on audits and ISO 9000 registration.

The Metrics group, including the Metrics Lead, was merged into the SQ group, which in turn was moved into the division responsible for the major voicemail product. It reported only to R&D for that division; no other corporate divisions retained any connection with the group. Furthermore, the U.S. Operations Vice President, a major champion of software quality within Telogics, was stripped of his major corporate responsibilities and became simply an administrator of the U.S. facilities.

A short time later, the CEO, the only former Techanswering executive in a major top corporate post, was moved into a ceremonial position. No Techanswering executives now occupied any top corporate posts. The focus of Telogics became more oriented to its Techvox legacy than its Techanswering legacy and became more marketing driven than engineering driven.

Establishing the Metrics Project

With the July reorganization, an executive who had been a key direct report for the DVP for R&D in The Division became the DVP for R&D in the new voicemail division. He became a strong champion for improving the predictability and hence the quality and timeliness of the Version C product. As long he remained a strong champion, the metrics project, as a part of the predictability effort, moved steadily forward.

The Version C predictability improvement effort started in June and accelerated with the corporate reorganization in July. Four teams were formed to develop plans for the effort: Metrics, Development Lifecycle, Defect Management, and Estimation. The SQ manager, the Metrics Lead, and an outside consultant developed a Metrics Plan, and the Metrics Team approved it, forming the foundation for the Metrics Program.

CASE DESCRIPTION

The Metric Profiles

The major basis for the metrics program was a set of Metric Profiles derived from templates in Augustine and Schroeder (1999) which formed a set of measurements from which the actual measures used in the effort were chosen. The profiles encompassed four areas: Quality, Functionality, Time, and Cost. The measures were designed to answer key organizational questions, following the Goal-Question-Metric approach (Basili & Rumbaugh, 1988) and the metrics planning process of Fenton and Pfleeger (1997). A sample Metric Profile is shown in the Appendix. Note that the measurement state-of-the-art continues to move forward, with approaches being developed such as Measurement Modeling Technology (Lawler & Kitchenham, 2003).

While The Division did not formally seek a CMMI Level 2 status, the metrics program held to the spirit of the Measurement and Analysis Key Process Area (KPA) for CMMI Level 2 (CMMI Product Team, 2001).

Which Managers Received Reports and How They Used Them

The metrics-reporting program started in mid-August 2000, beginning with weekly
reports. They evolved and expanded, eventually reaching several reports per week, through the delivery of Version C in May 2001.

Initially the report went to the Project Manager. However as time passed other people were added to the list, so that eventually it also went to the DVP of R&D, the director of testing, the two testing managers on Version C, the Version C development managers, and the two people maintaining test-case, defect-fix, and defect-verification data.

The project manager used the report in his weekly status reviews with the DVP. In addition he used the information as part of his presentation at the R&D Weekly Project Reviews, a management forum to communicate status for projects in R&D. Others in the division also used the report. For example, the director of testing used it in his meetings with the testing managers to make sure that the all their quality-related data were consistent.

Overall, the uses of the metrics by management were as follows:

By project management—To track defect-resolution and testing progress, to become alert to problems early, to manage development and testing resources, and to respond to senior management.

By senior management—To assess the health of the project both absolutely and in comparison to the other projects within the organization, to communicate with customers on product progress, to plan ahead for product delivery and follow-up efforts.

What Metrics Were Reported

1. Pre-test-stage defects

The initial period of metrics reporting, during the summer and early fall, focused on pre-test metrics, i.e., defects, comments, and issues arising from the system analysis, system design, and coding phases of the SDLC. These characteristics were reported according to a Telogics procedure called the “Document and Code Review Process for Version C” (2000). The procedure introduced a consistent yardstick for declaring whether a characteristic was a defect or not. Each characteristic was judged to be one of three possibilities (see Figure 3 for a sample tabulation):

Defect A defect, with a severity of either major or minor.
Comment Not a defect.
Issue Further investigation is needed to determine defect status.
To be resolved to either a “Defect” or a “Comment.”

<table>
<thead>
<tr>
<th>Week Ending</th>
<th>Reviews Held</th>
<th>Total Incidents</th>
<th>Comments</th>
<th>Issues</th>
<th>Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>8/12/00</td>
<td>5</td>
<td>41</td>
<td>25</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>8/19/00</td>
<td>9</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>8/26/00</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9/2/00</td>
<td>7</td>
<td>45</td>
<td>15</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>9/9/00</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9/16/00</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
2. Test-stage defects

The most active period of metrics reporting, during the fall and winter, focused on metrics relating to testing and defects found during test. The testing took place in four phases—preliminary (Phase 1), integration (Phase 2), full-scale or quality-assurance testing (Phase 3), and system testing (Phase 4). All of the test phases met their scheduled completion dates.

The defects were kept in a commercially available database called the Distributed Defect Tracking System (DDTS). Since DDTS was an Oracle-based system, databases such as Access could link to it, and thus automated metrics tracking systems became possible. The Metrics Lead used Access queries and programs extensively to obtain the defect data for the required reports (see sample chart in Figure 4), which made it feasible to produce presentations in a day. This allowed for the variety and volume of reports that management required, as well as reports based on real-time data often no more than an hour or two old.

Figure 4. Sample Defect Graph

3. Tests scheduled, executed, passed, failed, and blocked

The tracking of tests scheduled, executed, passed, failed, and blocked (Figure 5) was maintained by an internally developed test tracking system, which will herein be called TestMate. TestMate was developed using Visual Basic and Access, and included password-protected updating, identification of defects by feature sets within the products and by features within the feature sets, identification of DDTS defect numbers for tests finding defects, and identification of testers and developers.

Since TestMate was available Telogics-wide, it allowed anyone to see how testing was proceeding, either overall or for a particular feature. This provided an openness that
greatly aided the communication among the various organizations working on Version C and allowed the Metrics Lead to produce metrics reports in real time.

4. Code length and code changes

To put the number of defects and the testing effort in perspective, one needs to know the size of the software product. This is not as straightforward as would appear (Fenton & Pfleeger, 1997). The usual measure of size, the number of lines of code in the software, may be misleading. Some lines may be far more complex than others, and languages require different amount of lines to represent an action. More sophisticated approaches to code size than lines-of-code exist, such as function points or code complexity. However, these require more sophisticated tools and skills, neither of which Telogics had or was willing to acquire. Therefore, lines-of-code was indeed used to measure code size.

Even using the lines-of-code measure is not straightforward, however (Humphrey, 1995). While Version C had around two million lines of code, most of it was code carried over from previous versions. Only several hundred thousand lines were actually created, changed, or deleted from Version B (see Figure 6). The code counter had to include all builds within version C in order to provide a meaningful size metric.

![Figure 5: Sample Test Cases Table](image)

<table>
<thead>
<tr>
<th>Test Cases</th>
<th>12-Jan-2001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Cases</td>
<td>2,024</td>
</tr>
<tr>
<td>Pending</td>
<td>1,553</td>
</tr>
<tr>
<td>Executed</td>
<td>471</td>
</tr>
<tr>
<td>Passed</td>
<td>424</td>
</tr>
<tr>
<td>Failed</td>
<td>37</td>
</tr>
<tr>
<td>Blocked</td>
<td>21</td>
</tr>
</tbody>
</table>

![Figure 6: Sample Lines of Code Activity by Build](image)

<table>
<thead>
<tr>
<th>Build</th>
<th>Added LOC</th>
<th>Changed LOC</th>
<th>Deleted LOC</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32,428</td>
<td>3,851</td>
<td>1,889</td>
<td>38,168</td>
</tr>
<tr>
<td>2</td>
<td>1,746</td>
<td>11,385</td>
<td>472</td>
<td>13,603</td>
</tr>
<tr>
<td>3</td>
<td>167</td>
<td>0</td>
<td>0</td>
<td>167</td>
</tr>
<tr>
<td>4</td>
<td>283</td>
<td>84</td>
<td>0</td>
<td>375</td>
</tr>
<tr>
<td>5</td>
<td>106</td>
<td>37</td>
<td>36</td>
<td>181</td>
</tr>
<tr>
<td>6</td>
<td>481</td>
<td>96</td>
<td>20</td>
<td>577</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>77</td>
<td>9</td>
<td>136</td>
</tr>
<tr>
<td>8</td>
<td>2,329</td>
<td>797</td>
<td>382</td>
<td>3,508</td>
</tr>
<tr>
<td>9</td>
<td>1,496</td>
<td>1</td>
<td>0</td>
<td>1,497</td>
</tr>
<tr>
<td>10</td>
<td>7,913</td>
<td>10,195</td>
<td>1,473</td>
<td>19,581</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Total</td>
<td>87,462</td>
<td>40,111</td>
<td>25,772</td>
<td>153,395</td>
</tr>
</tbody>
</table>
The code size and activity information was very valuable:

- It provided a perspective of the defect exposure and risk, as well as determining whether a surge in defects was due to defect proneness or a high amount of activity.
- One could determine when development activity was basically complete, so that defect tracking and analysis no longer had a moving target and better estimates could be made.

5. Build, schedule, and cost reports

Part of the project planning included a build schedule (i.e., evolution of product functionality), which included all the builds to be created, the contents of each build, and the planned date for each build. As work proceeded and each build was created, the actual date was compiled and tracked against the planned date.

Another metric tracked was schedule. This was done through tracking major milestones for development, testing, and release. The milestone accomplishment dates were provided by the project manager who reported release timeline and content on a weekly basis.

The one significant schedule event was a slip for the beginning of full testing, due to delays in development for a critical feature, from September to December 2000. However, preliminary testing continued through this time on the significant portion of the project already developed, so by the time full testing formally began, much of it was already completed. The date for release testing therefore was met, as was the date for clean pass delivery and customer acceptance testing.

Cost was included at the start of the metrics effort. However, as the project proceeded, it became clear that cost was not a high priority for management, as compared to quality, functionality, and schedule. Hence early in the metrics effort, cost was dropped as a metric.

Evolution of the Metrics Reports

The reports evolved as time went by, with increasing acceptance and understanding, broader and higher distribution and exposure, ongoing requests for more information to be included, and a request from upper management for analyses in addition to reporting. By late in the project cycle, the reports covered a broad spectrum, including the following:

- Cumulative defects from review summaries and reviews held, by week (total and majors) and by lifecycle stage.
- DDTS defects found or fixed by week, by when found, and by product, projected date when defect criteria will be met, and defects found by function (development, testing, etc.).
- Estimated defects found (total and majors), estimated defects fixed, estimated defects remaining, projected defect count, estimated versus actual.
- Defects fixed but not verified, projected date when verification backlog will have been processed.
- Test cases to be run, executed, passed, failed, blocked (especially for one critical product), projected date when all tests will have been run, project date when tests-passed criteria will be met.
• Resources available and used (testers and fixers and verifiers).
• KLOCs by build (created, changed, deleted, total), etc.
• Analyses—evaluation of current data, alerts as to upcoming problems, de-alerts as to problems addressed, conjectured reasons for discrepancies and problems, projected defects outstanding, tests passed, fixes verified date projections, and likelihood of target dates being met.

This evolution was a natural process, moving ahead as the Metrics Lead became more familiar with the analysis process and the overall project, as the project and management users became more familiar with the reports and how they could be used in day-to-day administration and project guidance and planning, and as data providers became more familiar with the data to produce, and as automated tools were developed and used. Especially key was the request by management at a relatively early stage in the reporting evolution to have the Metrics Lead include analyses and recommendations.

Managerial and Other Impacts of the Metrics Program

Version C delivered on time, whereas Version B was late, and Version C had much fewer customer-visible defects, especially major ones, than was the case with Version B. Precise numbers are not available, but a good relative approximation is:

<table>
<thead>
<tr>
<th></th>
<th>Version C</th>
<th>Version B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open customer-visible major defects</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Open customer-visible minor defects</td>
<td>30</td>
<td>150</td>
</tr>
<tr>
<td>Delivery vs. schedule</td>
<td>On time</td>
<td>2 weeks late</td>
</tr>
</tbody>
</table>

Version C also had many fewer installation problems. This outcome likely helped to restore Telogics’ reputation with the three major customers whose relationship had been strained by the experience of Version B. This was a major positive impact for senior management, who had become worried about these customers. The following aspects of the metrics program were instrumental in having this happen:

* A much broader and deeper spectrum of metrics were reported.*

Before the metrics program only a few metrics were being reported such as defects, test cases, and broad schedule adherence. As a result of the metrics program, the metrics became broader and deeper. Defects before testing were reported, as well as those during and after testing. Defects were reported by feature set as well as overall and were also broken out by new defects versus forwarded open defects from previous versions. Code size was tracked over the full build cycle, with specific additions and changes noted. Both test cases and schedule adherence were broken down by feature set.

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
All of this richer information allowed more comprehensive and specific analyses, often pinpointing alerts in specific areas even when the defect level was OK on an overall basis. The implications for management were that they could identify defects at earlier stages in the SDLC when they were much less expensive to repair, as well as being much more able to control the stability of their code size and build patterns.

- **Important issues were detected before reaching crisis levels.**

As the metrics program and the software development continued and evolved, the Metrics Lead and others were able to detect and alert anomalies in important areas before they became far off course and reached crisis level. Examples of these were:

- One of the feature sets in Version C was a new function, in fact a key deliverable for this version. Several times during the project, an unusually large number of defects began occurring in this function. Within days after such a trend began, the Metrics Lead detected it and notified the project manager and the DVP, resulting in corrective action being taken. Upon delivery, this function—a subject of worry during the project planning and development—operated smoothly for the customer, a key to their satisfaction.
- Open major defects, a major phase exit criterion, were tracked and supported. On several phases the Metrics Lead’s projections of open majors at the intended milestone date showed that the number of open majors would exceed the exit criterion. Management was alerted and told where in the product the large numbers of open majors were so that testing and repair efforts were focused on those areas. As a rule the numbers of open majors then decreased, resulting in the milestone criteria being met. Version C met the criterion of zero (or very few) open majors at delivery, whereas Version B had not.
- As Telogics moved through the final testing phase before delivery, they had resolved what was originally a large backlog of open defects down to a reasonable level. However, the Metrics Lead noticed that many resolved defects had not yet been verified and thus could not be considered completed. In fact, the backlog of resolved-but-not-verified defects began to increase to substantial proportions. Project and senior management were notified, who began assigning development staff to assist in verifying these defects. After several weeks of these efforts, the substantial backlog had been fully addressed.

Both project and senior management could respond much faster and effectively to these developments, a significant boon for both levels of management.

- **Management was able to give emphasis to areas of the project where it did the most good.**

With more detailed reports and the inclusion of analyses, both project and senior management had information by which they could emphasize areas of the project where they were most effective. A critical example was a particular new feature that was key to Version C but had risk in developing; the metrics program provided specific defect, code
build, schedule, and testing information on this feature. This allowed management to use its scarce time more effectively in guiding the Version C development.

CURRENT CHALLENGES/PROBLEMS FACING THE ORGANIZATION

A Fatal Organizational Shift

The metrics program succeeded in what it set out to do, and the Version C product was much improved from its predecessor. However, despite the metrics program’s success, larger corporate and organizational issues adversely changed the atmosphere in which the metrics program, and indeed the overall corporate quality function, operated. This changed atmosphere first undermined the organization, SQ, in which the metrics program resided, then did the same with the metrics program itself. The loss of support became apparent during the last several months of the project. The metrics program ended with the delivery of Version C, with no continuation to future projects.

In retrospect this was a culmination of an evolution that began taking place well before the July 2000 reorganization and may have begun with the merger of the two legacy companies in 1998—the struggle between the Techanswering part and the Techvox part of Telogics for dominance. While the merger was technically one of equals, it soon became clear that the Techvox part of Telogics was politically the stronger entity.

This seriously affected the long-term prospects for metrics, process improvement, and software quality assurance (SQA) as a whole. The former Techanswering had been very much engineering oriented and relatively friendly to SQA, while the former Techvox was more finance oriented and relatively unfriendly to SQA. So long as economic times were good and the Techanswering influence on Telogics was still significant, the environment was friendly to SQA. However, as the Techvox influence became more and more pervasive, and as economic times began to decline, this environment became much less so.

Before the July 2000 reorganization, the R&D and service portions of Telogics were organized as a U.S.-based division and a foreign-based division. This allowed the Techanswering atmosphere to thrive in the engineering-oriented U.S.-based division. This division had a strong Corporate Quality organization, independent from the R&D organization, which had direct ties to a similar organization in the old Techanswering.

The July 2000 reorganization changed Telogics’ structure from a horizontal one to a vertical one. Each major product had its own organization, with a sales and marketing group, an R&D group, and other functions. It became up to each organization as to whether or what quality activities they would carry out. Significantly, these organizations were headed by marketing executives who now reported directly to top management, bypassing engineering entirely.

Paradoxically, in the short term the reorganization helped the metrics program. Before, metrics had been a part of Corporate Quality, thus a staff activity removed from the R&D effort. However, with the reorganization the DVP of R&D for Telogics’ largest product organization wanted a strong metrics activity and established it as part of an independent SQ group reporting directly to him. Metrics therefore became a line activity with direct influence within this organization.
However, the general atmosphere continued to become more metrics-unfriendly. A few months after the reorganization, the overall corporate leadership of Telogics changed. This removed all American influence from the top corporate levels, from which the support for quality, process improvement, and SQA as a whole had come, and eventually affected the R&D DVP’s support.

In the early fall of 2000, the DVP hired a new director of testing for the division. Despite the DVP’s earlier emphasis on quality and predictability, the man he hired did not favor this approach; he emphasized testing to ensure quality, with little or no overall process improvement or measurement to ensure product quality before the start of testing.

This represented a major turning point in the division climate. A major unanswered organizational question is why the R&D DVP, who had been such a strong champion for predictability through measurement and process improvement, hired a key subordinate with such an opposite outlook. This move did not appear to be a hiring error, but rather the beginning of a significant change in the senior manager’s own outlook, as he over a short time became far less sympathetic to what he had championed earlier. Whether he himself had a change of heart or whether the hire and the change of policy were forced on him by upper management is still uncertain, but in retrospect the fate of future metrics efforts and process improvement was sealed at this point.

The new testing director did not want to have a separate software-quality group reporting directly to the DVP. He influenced the DVP to abolish the SQ group and absorb its members into the testing organization. The metrics program was now under the management of a person who opposed such a program.

Early in the spring of 2001, the main testing of Version C was completed and final testing began. At that stage the project manager, likely emboldened by the revised organizational situation, attempted to exclude the Metrics Lead from the reporting process. The DVP intervened and demanded the reporting continue. However, shortly thereafter even that support diminished and disappeared. In the spring of 2001, Telogics began facing a business slowdown, part of the slowdown in the overall telecommunications sector. In a layoff shortly thereafter, nearly all personnel, including the Metrics Lead, associated with metrics and software quality were released.

The slowdown hit Telogics hard. In the summer of 2001, the stock price fell sharply and has not recovered to date. Earnings dropped significantly and further layoffs occurred. The entire Telogics division in which the metrics program took place was phased out in favor of the division making the other version of the voicemail product, a division and a product with origins in the old Techvox.

Questions for the Organization

Why, with the evidence of the metrics program’s success, did the division choose to eliminate it? This is a difficult question to answer explicitly; the managers directly involved might not consciously know, and even if they did, would probably consider it sensitive and proprietary even within Telogics. However, a number of hypotheses can be raised:

• Financially tough times were beginning, as the stock price drop eventually was to show. Telogics did what a lot of companies do in such circumstances: cut costs and programs, especially newer, more innovative ones.
• The division was one that was not going to be a part of the company’s dominant future. It produced one of two versions of Telogics’ legacy product. Only one version was likely to survive, and since the other version derived from the dominant wing of the company, it was becoming ever clearer that the division was going to eventually lose out. The DVP may have given up hope for the long-term future of the division, so was no longer oriented towards long-term thinking.
• The DVP may have been the sort of person who goes with the dominant paradigm, whatever it is at the time. At the beginning of the metrics effort, that paradigm was still quality and predictability, while at the end it had become strictly maintenance and legacy.
• The testing director was not on board when the metrics program was created, so he did not have any ownership in it.

The question for Telogics is how to assure quality and predictability of its products in the adverse times it is facing and to preserve the legacy of its metrics program. In these times the temptation is strong to cut all possible costs, spending only what is absolutely necessary to create and market its products and services. Moreover, many of the quality and metrics champions have either voluntarily or involuntarily left Telogics. However, even in adverse times customer satisfaction is very important. In fact, customer satisfaction is even more important in adverse times, since there are fewer customers and smaller orders, and a loss of a customer has much more impact and is harder to replace. The legacy of the metrics program may still be available for Telogics, both in terms of its physical artifacts and processes, and in terms of corporate memory. Since to ignore quality in these times is to invite a “death spiral,” this legacy is a valuable asset to Telogics, already paid for. However, this legacy is perishable, and the window of opportunity may be closing. The major challenges for Telogics are therefore:
A. How does Telogics best use this hard-earned legacy to assure quality as it continues to do business in these critical times?
B. How can Telogics best preserve the tools, artifacts, and data from the metrics program so that these assets can be easily recovered in the future?

Should Telogics consider a future metrics or other software process improvement program, they will need to address a number of questions arising from the metrics program of this case study. These questions apply not just to Telogics but also to other companies considering such efforts.
1. What degree of independence must a metrics program have, and how does one maintain that independence in the face of the natural inclination of a project manager and a testing director to want to control the formulation, compilation, and reporting of data? Note: there may be parallels here with the relationship of the Bureau of Labor Statistics to the rest of the Department of Labor.
2. How does one effectively obtain defects before the testing phase in the software development life cycle (SDLC)? How does one define a defect in the requirements, analysis, or design phases? How does one define severity, and how does one define when such a defect is corrected?
3. How does one effectively and feasibly carry out requirements traceability (the linking of software defects to the design elements, specifications, or requirements they affect)? This was one of the most difficult elements of the metrics program.

4. Quality, process improvement, and metrics efforts often have variable support. There are periods where strong support exists, especially when times are good. However, when times become difficult, too often the support wanes, and such programs become one of the first cutbacks.
   a. How do quality champions create an environment where quality and metrics efforts are faithfully followed in good times or bad?
   b. In order to make continuous overall progress in the variable environment, a company must generate strong advances during the favorable times, but then preserve them during the unfavorable times until support again emerges. Note that since during unfavorable times the personnel carrying out these efforts may likely leave voluntarily or be laid off, the preservation effort must survive such occurrences. What is the best way to make this happen?

5. What is the best way to effectively market a metrics program in an organization, both to get it established in the first place and to maintain its political support once it is in operation?

6. In the time span during which the metrics program operated, the Internet (and intranets and extranets) has become an increasingly important manner of communication. Any future metrics program should make full and effective use of the Internet. What is the best manner to use the Internet to collect metrics data, to calculate the metrics, to formulate effective reports, and to disseminate (active) and allow easy access (passive) to them?

7. How does one develop a win/win situation between the Metrics Lead and the Project Manager, that is, how can one construct the metrics effort so that its success is beneficial personally to the project managers and their careers? Similarly, how does one develop a win/win situation between the Metrics Lead and the Development Manager, as well as the Metrics Lead and the Testing Manager?

8. In relation to the previous question, a possible mechanism is to establish, through software process improvement organizations or industrial consortiums, a knowledge base of successful and unsuccessful metrics efforts (articles such as Goethart & Hayes (2001) represent a step in this direction). In this way if loss of support causes the loss of corporate memory and infrastructure for metrics, it is preserved within the community organizations for other enterprises to use, or even the original company when support once again arises. What is the best way to bring this about?

Covey, Merrill, and Merrill (1994) describe four quadrants of activities, as to whether or not the activities are urgent and whether or not they are important. They emphasize “Quadrant II activities”—those that are important but not urgent—stating that these tasks require continual oversight, lest they remain unaddressed because they are not urgent, but hurt the individual or organization because they are important. Software process improvement (SPI), in particular software measurement and analysis, is for most organizations a Quadrant II activity. Unlike testing, a close relative, it is not required in order to ship a software product. However, those organizations that neglect
SPI over the long haul suffer declines in the business worthiness of their software products, as product quality declines because of a lack of quality assurance and as product quality becomes more important as a product matures (Rothman, 1998). The experience of this case illustrates the challenge in establishing and maintaining an SPI as a Quadrant II activity, but also SPI's corresponding significant value to product quality, customer satisfaction, and customer retention when that challenge is met successfully.

FURTHER READING AND REFERENCES
Covey, S., Merrill, R., & Merrill, R. (1994). *First things first: To live, to love, to learn, to leave a legacy.* New York: Simon & Schuster.

**APPENDIX**

**Sample Metric Profile**

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Importance</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect Find Rate</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

**Details**

**Metric Description:**

There are pre-test defects detected during requirements, design, and coding, documented in the online notes file. There are test defects, identified by developers during the integration phase and by test personnel during the test and validation phases. There are also defects found in deployment. Test and deployment defects are documented in the defect-tracking system, with severity ratings from 1 (show stopper) to 5 (not a defect).

**Method of Capture:**

From planning to the start of integration testing, defects are maintained in the online notes file, though in the future defects could be entered into the defect-tracking system. From the start of integration testing through deployment, defects are maintained in the defect-tracking system. There is a procedure to capture data in the defect-tracking system to an Excel spreadsheet.

**Who Is Responsible for Capturing:** Software Quality

**When Captured (frequency):**

Biweekly until start of validation phase, then weekly until deployment.

**When Metric Capturing Ends:** One year after deployment.

**Who Is Responsible for the Metric Report:** Software Quality

**To Whom it Is Reported:** Project management

**Who Analyzes the Data:** Product and Project management, Quality (Testing, Software Quality, Process Quality), Development managers

**Additional Comments**

**Link to Project Goals:** Provides a way to better quantify the effectiveness of non-test methods in finding defects early in the development cycle. May provide timely insight into the status of the testing phase and may provide indicators as to when the software product is ready for release. A high incidence of errors after deployment indicates ineffectiveness of test and non-test methods and increases project cost.

**Issues:** Currently, early-stage defects reported in the online note file are reported inconsistently, both in terms of whether a defect is reported at all, and the language and format consistency in how they are reported. In order to obtain meaningful metrics, a process needs to operate on when and how early-stage defects are reported. Note that for later-stage defects, there is such a process using the defect-tracking system, which allows for meaningful defect metric in these stages.
BIOGRAPHICAL SKETCH

David Heimann is a Professor in the Management Science and Information Systems Department of the College of Management at the University of Massachusetts, Boston. His PhD is in Computer Science from Purdue University. He has held positions in government and industry, performing activities in reliability modeling, simulation, database management systems, probabilistic modeling, software analysis, and software process improvement. Among his publications are articles in various Proceedings of the Annual Reliability and Maintainability Symposium and a book-length article on system availability in Advances in Computers. In 1984 he received the Pyke Johnson Award for a paper of outstanding merit from the Transportation Research Board.