Understanding Software Metric Use

David Henderson is assistant professor of accounting in the College of Business at the University of Mary Washington (Fredericksburg, Virginia, USA). He can be reached at *dhender3@umw.edu.*

Steven D. Sheetz is associate professor of accounting and information systems in the Department of Accounting and Information Systems in the Pamplin College of Business at Virginia Tech (Blacksburg, Virginia, USA). He can be reached at *sheetz@vt.edu.*

Linda Wallace is associate professor of accounting and information systems in the Department of Accounting and Information Systems in the Pamplin College of Business at Virginia Tech (Blacksburg, Virginia, USA). She can be reached at *wallacel@vt.edu.*

In the early 1990s, the baggage claim system at Denver International Airport (Colorado, USA) was designed to automate baggage handling by using software to direct baggage contained in unmanned carts running on a track. Unfortunately, errors in the software that controlled the baggage claim system resulted in substantial cost overruns, delayed the opening of the new airport and eventually resulted in complete abandonment of the system. The Denver baggage claim project illustrates the catastrophic impact a failed software project can have on an organization.[1]

An effective software metrics program can help prevent software project failures, such as the Denver baggage claim project, by evaluating and monitoring project progress, thereby helping to identify problems before they worsen.[2] A software metric provides a quantitative indication of some attributes of software, such as size, complexity or quality. Examples of software metrics include function points, cyclomatic complexity and source lines of code. The potential of software metrics to increase control of the software development process naturally makes the appropriate use of software metrics a concern for IS auditors.[3] Without the appropriate use of software metrics, the software development process may be loosely controlled, thereby making it difficult for IS auditors to assess and monitor risk during software development.

Although software metrics can provide greater control over the software development process, resistance to them has resulted in inappropriate use and high failure rates for software metric initiatives.[4, 5, 6, 7] More than 80 percent of software metric initiatives fail within the first 18 months.[8] Even when software metrics are used, development teams often use them inappropriately.[9] For example, despite arguments from the research community about why source lines of code (SLOC) are a poor measure of software size, development teams still commonly use them to assess productivity and provide cost and schedule estimates.[10] The improper application of a software metric, such as SLOC, can quickly lead to project failure if it produces flawed estimates.

One possible reason for resistance to software metrics is that members of a development team may not perceive the advantages of using software metrics. Development teams must perceive software metrics as useful; otherwise, they may use them reluctantly and inappropriately.[11, 12]

Another potential reason for resistance to software metrics is that different groups involved in software metrics initiatives (managers, developers and metrics coordinators) use software metrics for different reasons, implying that they have different perceptions about software metrics. When groups have varying perspectives of a technology, organizations may experience difficulty developing, implementing and using the technology.[13] These differences in perception among different stakeholders on a software metrics initiative could lead to communication problems and, ultimately, resistance to use.

Resistance to software metrics initiatives should concern IS auditors. Strong opposition to software metrics can result in inappropriate or unenthusiastic use or even deliberate obstruction of software metrics initiatives. Inappropriate or unenthusiastic use, in turn, may result in a less-controlled and riskier software development process. Since software metrics mitigate risk and increase control of the software development process, IS auditors should ensure that development teams use software metrics appropriately and determine whether groups within development teams perceive the benefits of software metrics differently.

Motivated by these issues, 126 managers, developers and metrics coordinators were surveyed to determine whether they understand the benefits of using software metrics and whether they perceive the benefits of software metrics differently. The results suggest managers, developers and metrics coordinators may not fully appreciate the benefits of software metrics and also indicate that these three groups perceive the benefits of software metrics differently.

IS AUDITORS AND SOFTWARE METRICS

Software development is fraught with risk, including variations in project scope, time overruns, cost overruns and inappropriate

resourcing/staffing model management. Software metrics can help mitigate the risk by serving as effective monitoring tools, thereby helping to mitigate risk and increase control of the software development process. For example, software metrics, such as function points, can help management plan software development projects, allocate resources, monitor software project progress, and watch for schedule and cost overruns.[14] Other software metrics, such as tracking defects per line of code, can help development teams ensure high software quality.[15]

The potential of software metrics to mitigate risk during the software development process, coupled with the IS auditor's responsibility to ensure that the development process is timely and cost-effective, makes the appropriate use of software metrics a concern for IS auditors. If development teams fail to use metrics appropriately, either because they fail to appreciate the benefits of metrics or because groups within the development team perceive the benefits of metrics differently, the development process may be loosely controlled. Accordingly, IS auditors should ensure that key software metrics have been established to measure the performance of the project team and the project and then take steps to ensure that metrics are used appropriately throughout the project.[16, 17] Furthermore, IS auditors should review service level agreements (SLAs) to determine if they utilize metrics that are monitored and measured.[18]

SURVEY DESIGN

To develop a framework for understanding whether managers, developers and metrics coordinators understand the benefits of using software metrics, prior research was reviewed to uncover the characteristics (i.e., the desirable properties) of effective software metrics. **Figure 1** lists the desirable properties of software metrics.[19]

This list of desirable properties formed the basis for the web-based survey. When completing the survey, respondents were asked to identify a software metric with which they were familiar and then respond to the questions developed from the desirable properties about that software metric. Survey

Figure 1—Desirable Properties of Software Metrics

Desirable Properties of Software Metrics	Definition	No. of Questions
Automatibility	The degree to which the collection of data for the metric and the metric's calculation are computerized	4
Calculation ease	The degree to which the value of the metric is easy to calculate	3
Data availability	The degree to which the data required to calculate the metric are readily available given the products and processes currently used	3
Intuitiveness	The degree to which the metric's behavior conforms to intuition	1
Language independence	The degree to which computation of the metric does not depend on the programming language used	3
Life cycle applicability	The degree to which the metric can be applied throughout the SDLC	4
Normativeness	The degree to which there is a standard, typical or normal range of "acceptable" values for the metric	3
Predictiveness	The ability of the software metric to estimate an important attribute to be realized in the future; for management metrics, the ability of the metric to provide accurate software size and effort estimates; for quality metrics, the ability of the metric to predict software quality	1
Prescriptiveness	The ability of the software metric to not only diagnose problems, but suggest solutions; for management metrics, the ability of the metric to help diagnose problems in the software development process and make changes accordingly (e.g., increase resources to improve schedule performance); for quality metrics, the ability of the metric to help diagnose problems in software quality and recommend solutions accordingly	4
Sensitivity	The degree to which the metric is sensitive to changes in the attribute(s) measured	1
Timeliness	The degree to which the metric provides feedback in time to affect the outcome	1
Understandability	The degree to which the metric is easy to understand; the degree to which the metric is free of mental effort	2
Validity	The degree to which the software metric assesses the attributes it purports to measure; the degree to which it has been empirically tested and supported	4

question responses were measured on a five-point rating scale ranging from one to five, in which one equaled strongly disagree, three equaled undecided and five equaled strongly agree. **Figure 2** lists the survey items.

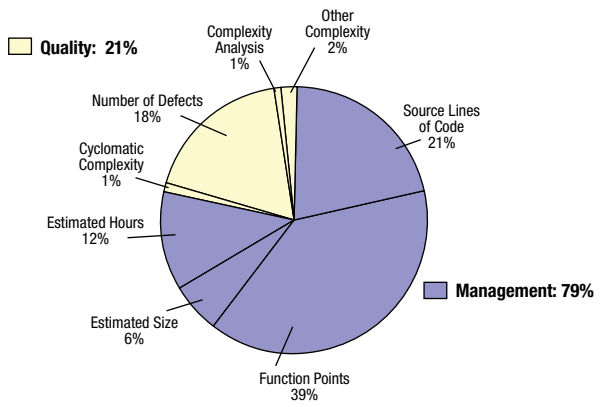
Data were collected from three different sources over a six-month period. The first source consisted of members of a computer software metric Usenet group. The second source included members of the Information Systems Special Interest Group of the Project Management Institute (PMI-ISSIG). Additional participants were employees of a large IT consulting company. The final sample consisted of 126 managers, developers and metrics coordinators.

Figure 2— Survey Items

Desirable Property	Survey Item
Automatibility	The data required to calculate the measure can be automatically collected.
Automatibility	The measure can be calculated by a computer program.
Automatibility	A computer program can interpret the measure.
Automatibility	Data collection and calculation of the measure can be automated.
Calculation ease	The calculation of the measure is straightforward.
Calculation ease	The measure is easy to calculate.
Calculation ease*	Calculating the measure is often frustrating.
Data availability	The data required to calculate the measure are readily available in the current software development environment.
Data availability	Analysis of the measure can be performed using data from the existing software development process.
Data availability	The measure can be calculated without having to collect additional data.
Intuitiveness	The measure behaves according to intuition.
Language independence	The measure is programming-language-independent.
Language independence	The choice of programming language does not affect the ability to calculate the measure.
Language independence	The calculation of the measure is not affected by the differences in programming languages.
Life cycle applicability	The measure can readily be used throughout the entire development process.
Life cycle applicability	The measure can easily be used repeatedly throughout a development process.
Life cycle applicability	The measure can support development in early and later stages.
Life cycle applicability	The measure can be easily applied to designs, specifications and software.
Normativeness	The measure has a well-known range of acceptable values.
Normativeness	The measure has established standards that can be used to interpret measured values.
Normativeness	A standard range of values for the measure is known.
Predictiveness	The measure improves the ability to predict success.
Prescriptiveness	The measure improves the ability to identify problems with the software.
Prescriptiveness	The measure makes it easier to identify methods for improving the software.
Prescriptiveness	The measure increases the ability to identify new procedures that should be followed.
Prescriptiveness	It is easier to solve problems when using the measure.
Sensitivity	The measure is highly sensitive to changes in the software.
Timeliness	The measure can be used in time to improve the software.
Understandability	The measure is easy to understand.
Understandability	The use of the measure requires little mental effort.
Validity	The measure has been rigorously tested in the field.
Validity	The measure has been extensively empirically validated.
Validity	The measure is highly credible.
Validity	The scale of the measure is appropriate.
*Denotes reverse-coded item	

The software metrics identified by the respondents were categorized as either management or quality metrics, and the data analysis was conducted along those dimensions. Software metrics typically used to control the software development process, such as function points and SLOC, were classified as management metrics. Software metrics used to monitor software quality, such as cyclomatic complexity and number of defects, were classified as quality metrics. **Figure 3** shows the software metrics identified by the participants, their respective categorization as quality or management, and the percentage of respondents who identified each software metric.

Figure 3—Software Metrics Identified by Type



After categorizing the metrics identified by the survey respondents as quality or management, the job codes provided by each participant as manager, developer or metrics coordinator were classified. **Figure 4** shows position titles identified by the respondents and their subsequent mapping to the manager, developer and metric coordinator categories, along with the percentage of respondents they represent. The average respondent was 44 years old with 19 years of experience in the software industry and had used software metrics for 5.8 years. Of the 126 participants, 62 were managers, 45 were developers and 19 were metrics coordinators. Managers, developers and metrics coordinators had approximately the same amount of experience and were similar in age.

RESULTS

The average scores for each desirable property for each metric were then analyzed. **Figure 5** presents the average scores for each question for quality metrics. **Figure 6** presents the average scores for each desirable property for management metrics.

As indicated by the scores in **figure 5**, the average scores for all groups for all desirable properties except one (intuitiveness) are slightly higher than three (undecided), suggesting that while metrics coordinators, developers and managers generally perceive the value of quality metrics, they do not overwhelmingly believe in the value of quality metrics. Metrics coordinators generally have the most favorable perceptions

Figure 4—Mapping of Job Categories to Manager, Developer and Metric Coordinator Groups

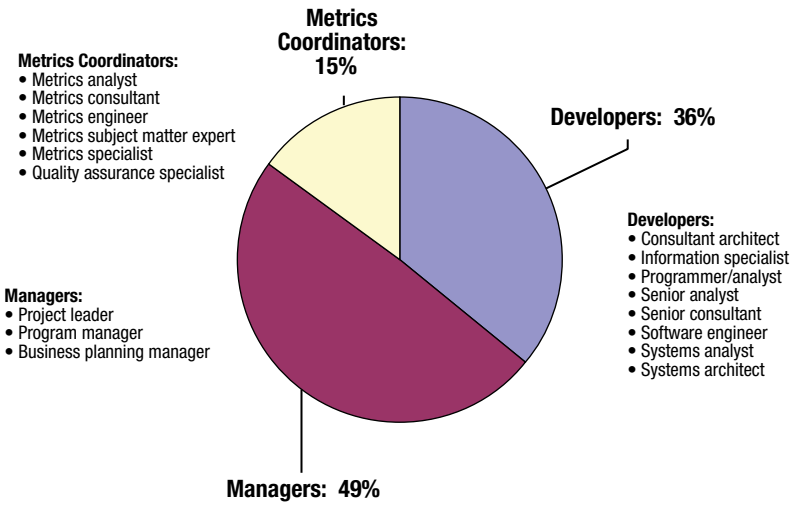


Figure 5—Average Scores by Desirable Property for Quality Metrics				
Property	**All Job Codes**	**Metrics Coordinators**	**Developers**	**Managers**
Automatibility	3.80	4.00	3.81	3.72
Calculation ease	3.91	4.27	3.70	3.92
Data availability	3.80	4.00	3.81	3.72
Intuitiveness	**2.89**	**2.92**	**2.89**	**2.92**
Language independence	4.11	4.53	3.93	4.08
Life cycle applicability	4.13	4.50	4.19	3.94
Normativeness	**3.49**	3.93	3.52	**3.31**
Predictiveness	4.15	4.00	4.11	4.23
Prescriptiveness	3.99	4.15	4.08	3.87
Sensitivity	3.67	4.40	**3.33**	3.62
Timeliness	4.22	4.60	4.11	4.15
Understandability	3.67	3.60	3.61	3.73
Validity	3.83	4.15	3.75	3.77
Average	3.82	4.08	3.76	3.77
Bold=below 3.5				

Figure 6—Average Scores by Desirable Property for Management Metrics				
Property	**All Job Codes**	**Metrics Coordinators**	**Developers**	**Managers**
Automatibility	**3.21**	**3.34**	**2.98**	**3.35**
Calculation ease	3.52	3.90	**3.48**	**3.44**
Data availability	3.68	4.19	3.63	3.56
Intuitiveness	**2.96**	**3.07**	**3.08**	**2.84**
Language independence	3.53	**3.24**	4.08	**3.21**
Life cycle applicability	3.99	4.32	4.16	3.78
Normativeness	3.65	3.86	3.62	3.62
Predictiveness	3.79	4.14	3.97	3.55
Prescriptiveness	**3.23**	**3.25**	**3.38**	**3.12**
Sensitivity	3.55	4.36	**3.11**	3.63
Timeliness	3.62	3.86	3.75	**3.45**
Understandability	**3.32**	3.79	**3.18**	**3.29**
Validity	3.91	4.23	3.92	3.81
Average	3.54	3.81	3.57	**3.43**
Bold=below 3.5				

of quality metrics, as they have the highest average scores for every desirable property, except for understandability and predictiveness. It is surprising that developers did not perceive quality metrics more favorably, given that these metrics are used to monitor software quality. All three groups indicated that quality metrics are not intuitive (average is less than three), suggesting that training programs may be needed in the beginning of a software metrics implementation.

As shown in **figure 6**, no scores for any desirable property for all job codes exceeded four (agree), indicating that these groups, as a whole, do not completely appreciate the benefits of management metrics. Similar to the findings for quality metrics, metrics coordinators typically have more favorable views of management metrics than managers. Unlike the findings for quality metrics, developers perceive higher value of management metrics than managers. This result is counterintuitive given

that managers should be more reliant on management metrics than developers, as these metrics are used for monitoring productivity. The average scores for management metrics for the prescriptiveness desirable property are slightly higher than three (undecided) for all groups. This finding is surprising given that management metrics should be useful for diagnosing problems in the software development process (e.g., increase resources to improve schedule performance); thus, prescriptiveness should be a main reason for using management metrics. Furthermore, management metrics are not perceived as intuitive, suggesting that additional training on management metrics may be useful.

IMPLICATIONS AND RECOMMENDATIONS FOR IS AUDITORS

These results indicate that managers, developers and metrics coordinators may not fully understand the benefits of management metrics. Management metrics, such as function points, should help managers, developers and metrics coordinators recognize problem areas in the software and suggest solutions accordingly. Thus, it would be expected that all three groups would use management metrics for their prescriptiveness. Interestingly, however, all three groups were undecided about the prescriptiveness of management metrics. This finding should be a concern for IS auditors as it suggests that managers, developers and metrics coordinators may not fully appreciate the benefits of using management metrics, which, in turn, may cause inappropriate use of software metrics.[20, 21] Furthermore, all three groups seem to perceive quality metrics more favorably than management metrics. Given that management metrics, such as function points, should be established to monitor and control the systems development process,[22] IS auditors should, therefore, ensure that members of a development team appreciate the value of software metrics. IS auditors can accomplish this task via observation, inquiries, and taking an active, yet independent, role in the systems development process.[23]

Previous studies on software metrics have found that managers perceive software metrics as more useful than developers.[24] On the contrary, this survey found that developers and metrics coordinators better understand the benefits of management metrics, more so than managers. Given that managers use management software metrics, such as function points to provide software size estimates, it would be expected that managers, more so than developers, use management metrics to predict level of effort and software size. Results, however, indicate that managers do not use management metrics for their predictiveness or prescriptiveness. This finding is a concern for IS auditors as it suggests that managers may not understand the value of management metrics, which could lead them to use software metrics inappropriately. Using software metrics inappropriately, in turn, can hinder the ability to control the software development process and mitigate risk. IS auditors should, therefore, ensure, via observation and interviewing techniques, that management understands the value of software metrics and is using the appropriate software metrics.

Survey results also show that managers, developers and metrics coordinators perceive the benefits of software metrics differently. As mentioned previously, metrics coordinators and developers appear to have more favorable perceptions of software metrics than managers. IS auditors should be aware that these groups perceive software metrics differently and that these differences in perceptions can lead to opposition against software metrics. IS auditors should monitor the perceptual differences between these groups and ensure that these differences in perception do not result in inappropriate use. Accordingly, IS auditors should take an active role in the software development process and ensure that effective communication between different groups on the development team is occurring and that each group appreciates the values of the other groups.

CONCLUSION

To raise awareness of the benefits of software metric use and potentially improve communication among managers, developers and software metrics coordinators during software metrics initiatives, organizations should develop integrated and comprehensive education and training programs. Education and training efforts should include an overview of the potential benefits to all groups, followed by a focus on those characteristics of using software metrics tailored to each group. For example, metrics coordinators do not use management metrics for a range of issues important to managers and developers, including predictiveness; thus, metrics coordinators could, instead, be instructed on the predictiveness benefits of management metrics. Additionally, members of all three groups may benefit from training in communication techniques that emphasize understanding the views of others and communicating using the target audience's concepts and terminology. IS auditors can play a

role in these education and training efforts by ensuring that metrics coordinators, developers and managers have received adequate training on the benefits of using software metrics.

Another strategy for increasing the effectiveness of software metrics initiatives is to use a dedicated metrics team to facilitate the implementation of software metrics programs.[25, 26] These survey results indicate that metrics coordinators perceive the value of metrics more than managers and developers. Metrics coordinators, who serve as liaisons between managers and developers during software metrics initiatives, are particularly well positioned within the organization to fill this role and, hence, can help managers and developers better understand the benefits of software metrics. Further, IS auditors should ensure that the organization has a dedicated metrics team to assist with the implementation of software metrics initiatives.

Although implementing education and communication programs may improve awareness of the benefits of software metrics and potentially increase use, it may be that more effective software metrics are needed with clearly defined goals. This survey indicates that quality metrics are not used for their ability to identify problems with software quality, potentially implying that managers, developers and metrics coordinators believe prescriptive quality metrics simply do not exist. Perhaps efforts should not only be directed toward education and training, but also toward developing software metrics that more practitioners perceive as predictive and prescriptive.

ENDNOTES

1 Calleam Consulting, "Case Study—Denver International Airport Baggage Handling System—An Illustration of Ineffectual Decision Making," 2008, *http://calleam.com/WTPF/?page_id=2086*

2 Ewusi-Mensah, K.; "Critical Issues in Abandoned Information Systems Development Projects," *Communications of the ACM*, vol. 40, iss. 9, 1997, p. 74-80

3 ISACA, *CISA Review Manual 2014*, USA, 2013, *www.isaca.org/bookstore*

4 Rubin, H.; "Measure for Measure," *Computerworld*, vol. 25, 15 April 1991, p. 77-78

5 Roche, J.; M. Jackson; M. Shepperd; "Software Measurement Methods: An Evaluation and Perspective," 3rd Symposium on Assessment of Quality Software Development Tools, Washington DC, USA, June 1994

6 Jones, C.; *Applied Software Measurement: Assuring Productivity and Quality*, McGraw-Hill, 1991

7 Gopal, A.; M. S.Krishnan; T. Mukhopadhyay; D. R.Goldenson; "Measurement Programs in Software Development: Determinants of Success," *IEEE Transactions on Software Engineering*, vol. 28, iss. 9, 2002, p. 863-75

8 *Op cit*, Rubin

9 *Ibid.*

10 Fenton, N. E.; Martin Neil; "Software Metrics: Successes, Failures and New Directions," *Journal of Systems and Software*, vol. 47, iss. 2 and 3, 1999, p. 149-57

11 Hall, T.; Neil Fenton; "Implementing Effective Software Metrics Programs," *IEEE Software*, vol. 14, iss. 2, 1997, p. 55-65

12 Grady, R. B.; *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, 1992

13 Orlikowski, W.; Debra Gash; "Technological Frames: Making Sense of Information Technology in Organizations," *ACM Transactions on Information Systems (TOIS)*, vol. 12, iss. 2, April 1994, p. 174-207

14 *Op cit*, ISACA, 2013

15 *Ibid*.

16 *Ibid*.

17 ISACA, *Systems Development and Project Management Audit/Assurance Program*, 2009, *www.isaca.org/auditprograms*

18 *Ibid*.

19 Henderson-Sellers, B.; *Object-Oriented Metrics: Measures of Complexity*, Prentice-Hall, 1996

20 *Op cit*, Hall and Fenton

21 *Op cit*, Grady

22 *Op cit*, ISACA, 2009

23 Henderson, D.; "Issues With Auditing the Systems Development Process," *ISACA Journal*, vol. 6, 2008, p. 42, *www.isaca.org/journal*

24 Sheetz, S. D.; D. Henderson; L. Wallace; "Understanding Developer and Manager Perceptions of Function Points and Source Lines of Code," *The Journal of Systems & Software*, vol. 82, iss. 9, 2009, p. 1540-9

25 *Op cit*, Hall and Fenton

26 *Op cit*, Grady