2005

Understanding Manager and Developer Perceptions of the Relative Advantage, Compatibility, and Complexity of Function Points and Source Lines of Code

Steven Sheetz
Virginia Tech, sheetz@vt.edu

Linda Wallace
Virginia Tech, wallacel@vt.edu

David Henderson
Virginia Tech, davidlhenderson@vt.edu

Understanding Manager and Developer Perceptions of the Relative Advantage, Compatibility, and Complexity of Function Points and Source Lines of Code

Steven Sheetz
Virginia Tech
sheetz@vt.edu

David Henderson
Virginia Tech
davidlhenderson@vt.edu

Linda Wallace
Virginia Tech
wallacel@vt.edu

ABSTRACT

Software measures are recommended for the effective management of software development projects. Innovation diffusion theory (IDT) provides perspective for understanding managers' and software developers' perceptions of the relative advantage, complexity, and compatibility of software measures. This paper describes the results of a survey in which software developers and managers identified a software measure and then answered IDT-based questions about the measure. Two of the most commonly identified measures were source lines of code (SLOC) and function points (FP). Overall, participants indicated that FP have greater relative advantage, compatibility, and complexity than SLOC. Developers indicated that FP have greater relative advantage, compatibility, and complexity than SLOC. Managers, however, did not perceive a significantly greater relative advantage and compatibility for FP over SLOC, but did perceive FP to be more complex than SLOC.

Keywords

Software Measures, Function Points, SLOC, Practitioners, Innovation Diffusion, Survey

INTRODUCTION

A software measure, or metric, is any tool that provides a quantitative indication of some attribute of software such as size, complexity or quality. Measures provide insight that enables project managers or software developers to improve the development process, or the software itself. Function points (FP) and source lines of code (SLOC) represent the two most common software measures for estimating software size and monitoring programmer productivity (Fenton & Neil, 1999; Heiat & Heiat, 1997; Kemerer, 1993).

Researchers have often encouraged practitioners to use FP to achieve improvements in software development processes. Yet, practitioners still commonly use SLOC (Fenton & Neil, 1999). This study evaluates practitioner perceptions of FP and SLOC along the innovation diffusion theory (IDT) dimensions of relative advantage, compatibility, and complexity to understand why SLOC continue to be used extensively.

Hall and Fenton (1997) describe a "troublesome gap" between developer and manager perceptions of software measures. If developers fail to perceive the advantages of software measures, they may not accept them (Fenton & Neil, 1999). Similarly, poor decisions can result if managers fail to understand a software measure or developers' perceptions of the measure. Thus, this study attempts to understand whether a gap in perception exists between developers and managers regarding SLOC and FP.

This paper is organized as follows: Section II provides a synthesis of prior research on FP and SLOC and suggests how IDT could explain the adoption of either measure, Section III discusses the research methodology, Section IV discusses the results, Section V provides a discussion of the results and implications for software development practice and Section VI offers concluding remarks.

BACKGROUND

Function Points (FP) and Source Lines of Code (SLOC)

FP and SLOC represent the two most commonly used software sizing and productivity monitoring software measures (Fenton & Neil, 1999). FP are designed to measure system size by quantifying the amount of functionality provided to the user in terms of the number of inputs, outputs, and files (Albrecht & Gaffney, 1983; Kemerer, 1993; Orr & Reeves, 2000). SLOC is a classic and simple software measure used to measure the amount of code in a program. FP and SLOC constitute competing substitutes for estimating software size and for monitoring programmer productivity (Kemerer, 1987; Ray, 1993). Although prior academic research maintains that FP are superior for estimating software size and for monitoring programmer productivity, SLOC are still commonly used (Fenton & Neil, 1999).

Prior Research on Perceptions of Software Measures

Although prior research has discussed the advantages and disadvantages of FP and SLOC, few researchers have examined practitioner perceptions of software measures. Hall and Fenton (1997) examined software developer and project manager perceptions of software measurement programs, without investigating specific measures. The authors found that managers were generally more positive about using measures than developers. This study digs deeper than the Hall and Fenton (1997) study by examining practitioners' perceptions of two competing software sizing and productivity monitoring measures—FP and SLOC—using an IDT-grounded framework.

Innovation Diffusion Theory and the Desirable Properties of Software Measures

Rogers (1983) defines an innovation as any product or process which is perceived to be new by a potential adopter. A software measure constitutes an innovation for many developers and managers. Without the use of software measures, managers must control the complexity of the software development process "by the seat of their pants". Software measures allow managers to introduce a new level of rigor into the software development process. The replacement of haphazard techniques that are congruent with the culture of software developers is necessary to mitigate software development cost and schedule overruns. Furthermore, given the purported advantages of FP over SLOC, practitioners should perceive FP as an innovation.

IDT states that several factors affect the adoption of an innovation: relative advantage, compatibility, complexity, trialability, and visibility. Research has determined that the first three factors are the most important in terms of explaining the adoption of many innovations. Rather than predicting the adoption of software measures, we apply these constructs as a framework to understand the perceptions of practitioners regarding FP and SLOC. Thus, issues of volitional control of the adoption of the measure are not relevant to the current study. However, the IDT constructs provide a valid framework for understanding practitioner perceptions of FP and SLOC.

In the software measurement literature, researchers have proposed various properties that software measures should possess. Henderson-Sellers (1996) provides a useful summary of the desirable properties of software measures. These desirable properties of software measures have been mapped into the IDT constructs as shown in Table 1.

Relative advantage is the perceived net value of an innovation. The more the perceived benefits exceed the perceived costs, the more likely the innovation will be adopted. In the context of FP and SLOC, relative advantage is the ability of the software measure to provide more accurate size estimates and monitor programmer productivity.

The literature suggests that FP are more applicable throughout the entire life cycle than SLOC. FP can be used continuously throughout the entire systems development life cycle (SDLC), whereas SLOC can only be calculated during the coding phase (Low & Jeffrey, 1990; Jones, 1991). Not only can FP be estimated early in the life cycle, they can also be recalculated as development continues (Fenton & Pfleeger, 1996).

Research also suggests that FP are more normative than SLOC. FP employs a normative standard for calculation that obtains FP estimates through a formalized set of procedures. FP are supported and endorsed by an independent standards body, the International Function Point User Group (IFPUG). On the contrary, SLOC lacks a normative, standardized counting process (Armour, 2004; DeMarco, 1982; Low & Jeffrey, 1990).

Prior literature also suggests that FP are more predictive than SLOC. FP are superior to SLOC for software size estimation purposes because they provide managers with accurate a-priori estimates of software size (Low & Jeffrey, 1990).

The literature suggests that FP are more prescriptive than SLOC (Fenton, 1996). For example, FP can be used to identify scope creep by repeating estimates at major milestones in the development process. Similarly, FP are a better productivity monitoring tool because FP, unlike SLOC, can be compared across languages. Thus, managers can determine if a tool, language or environment is more productive when compared with others (Albrecht & Gaffney, 1983; Low & Jeffrey, 1990; Ray, 1993). By identifying unproductive elements of the software development process and monitoring scope creep, FP allow managers to better diagnose problems and suggest solutions.

Software measures must also be sensitive to changes in the software to provide the necessary feedback to adjust processes. As the size of the software increases through additional requirements, SLOC and FP will also increase. Thus, SLOC and FP should both be sensitive to changes in the software.

Desirable Properties of Software Measures	**Definition**
Relative Advantage	
Life Cycle Applicability	The degree to which the measure can be applied throughout the SDLC.
Normativeness	The degree to which there is a standard, typical, or normal range of "acceptable" values for the measure.
Predictiveness	The ability of the software measure to estimate an important attribute to be realized in the future.
Prescriptiveness	The ability of the software measure to not only diagnose problems but suggest solutions.
Sensitivity	The degree to which the measure is sensitive to changes in the attribute(s) measured.
Timeliness	The degree to which the measure provides feedback in time to affect the outcome.
Validity	The degree to which the software measure assesses the attributes it purports to measure. The degree to which it has been empirically tested and supported.
Compatibility	
Data Availability	The degree to which the data required to calculate the measure are readily available given the products and processes currently used.
Language Independence	The degree to which computation of the measure does not depend on the programming language used.
Complexity	
Automation	The degree to which the collection of data for the measure and the measure's calculation and interpretation are computerized.
Calculation Ease	The degree to which the value of the measure is easy to calculate.
Intuitiveness	The degree to which the measure's behavior conforms to intuition.
Understandability	The degree to which the measure is easy to understand. The degree to which the measure is free of mental effort.

Table 1: Constructs associated with relative advantage, compatibility, and complexity of software measures

Timeliness is also an important property of a software measure if the measurements provided are to be used to guide projects. SLOC, however, it is not timely because it cannot be calculated until the coding phase (Dolado, 1997; Fenton & Pfleeger, 1996; Kemerer, 1992; Moser, Henderson-Sellers, Misic, 1999). FP are substantively more timely than SLOC because they can be computed earlier in the SDLC than SLOC.

Validity is often mentioned as a necessary property for successful measures (Henderson-Sellers, 1996). FP have been criticized as unreliable because they depend on subjective estimates rather than objective data (Mahmood, Pettingell, Shaskevich, 1996). Although SLOC represents an objective measure because it is calculated by the compiler, estimating SLOC before the coding phase requires expert judgment. (Matson, Barret, Mellichamp, 1994). Hence, its reliability and validity as an estimate of software size is dependent on the experience of the analyst.

In sum, the literature suggests that FP exhibit greater life cycle applicability, normativeness, predictiveness, prescriptiveness, and timeliness than SLOC. Both measures appear to exhibit sensitivity to changes, and both have validation issues. Overall, however, the literature indicates that FP have a relative advantage over SLOC.

Compatibility represents the degree to which an innovation is perceived as consistent with existing values, past experiences, and current needs. Compatibility is important for adoption for two reasons. First, the more an innovation intersects with existing values and beliefs, the more likely it will be adopted. Second, the more an innovation fits with existing practices, the fewer the startup difficulties and conversion costs involved in adoption.

The literature suggests that the data required to calculate SLOC are more readily available than the data required to calculate FP. FP measures must be extracted from design documents; SLOC can be easily calculated from any programming environment that contains code (Moser et al., 1999).

FP are considered more programming language independent than SLOC (Albrecht & Gaffney, 1983; Low & Jeffrey, 1990). SLOC essentially penalizes higher-level languages; programmers who code in higher-level languages appear less productive than programmers who code in lower-level languages (Jones, 1991).

Overall, the literature suggests that FP are more programming language independent than SLOC, yet the data required to calculate SLOC are more readily available than the data required to calculate FP. Thus, no firm conclusions can be drawn regarding the compatibility construct from our analysis of previous research.

Complexity represents the degree to which an innovation is difficult to understand and use. Less complex innovations are easier to understand, are easier to implement and less frustrating to use; thus, all else being equal, the lower the perceived complexity, the more likely the innovation will be adopted.

FP are more difficult to automate and more complex to calculate than SLOC. SLOC can be easily calculated and do not require subjective counting procedures (Fenton & Neil, 1999). The procedures for counting FP are labor-intensive, require expertise and experience, and do not lend themselves to automatic data collection (Heiat & Heiat, 1997; Kemerer, 1993; Moser et al.., 1999; Tate & Verner, 1991). SLOC are also more intuitive than FP. Unlike FP, SLOC can be easily visualized; for example, developers and managers alike can easily visualize 1000 lines of code. FP, on the other hand, are more difficult to visualize (Fenton & Pfleeger, 1996).

Given that SLOC can be more easily calculated than FP, that the calculation of FP cannot be easily automated, and that SLOC are more intuitive than FP, the literature suggests that SLOC are less complex than FP.

RESEARCH METHODOLOGY

Instrument Development and Validation

In the first stages of the research project, the authors examined the literature to identify the desirable properties of software measures. The desirable properties defined by the existing literature are presented in Table 1.

A survey instrument was developed that included items for each desirable property. The participants addressed the items using a 5-point likert scale ranging from strongly disagree to strongly agree. These items were integrated into a comprehensive list and changes were made until agreement was reached on the wording and meaning of each item. The items were then randomly placed into a document and 20 professors were asked to group similar items together. The goal of this step was to provide an initial check into the face validity of the categories and item measures. Any items not placed with the others in their category by at least 70% of the respondents were candidates for revision or deletion.

In the next phase, demographic items were added to the survey. The survey was pre-tested with 15 individuals who had experience with software measures and changes were made to the survey as a result of their feedback. When completing the survey, respondents were asked to identify software measures with which they were familiar. From the list they created, they were asked to select one measure and describe how the measure should be used. Subjects were then asked to assess the measure by responding to the items developed from the desirable properties on a scale ranging from strongly disagree to strongly agree.

Data collection involved three groups of subjects beginning in late 2003 and ending in summer 2004. The first group consisted of members of the computer software measurement Usenet group. An e-mail was posted to the group asking them for assistance in completing a web-based survey. The second source was individuals who are members of the information systems special interest group of the project management institute (PMI-ISSIG). The members of this group were sent an e-mail from the ISSIG coordinator asking them to participate in the web-based survey. Additional participants were obtained from members of a software measures user group at a large consulting company, yielding a total of 184 responses to the survey. Of the 184 responses, 70 subjects selected FP or SLOC. Therefore, the analysis presented in this paper is limited to these 70 responses.

RESULTS

Seventy of the participants who completed the survey identified either FP or SLOC; of these 70 participants, 28 were developers and 42 were managers. To investigate developer and manager perceptions of FP and SLOC, several analyses were conducted. First, the perceptions of the developers and managers for FP and SLOC for the constructs of relative advantage, compatibility, and complexity were compared across all participants. Second, IDT constructs and associated desirable properties measures were used to compare developers' perceptions of FP to SLOC, and to compare managers' perceptions of FP and SLOC. Third, perceptions of developers and managers were compared for FP and SLOC. All comparisons used independent sample t-tests.

The average respondent was 44 years old with 19 years in the software industry and had used software measures for 5.4 years. Developers and managers had about the same amount of experience and were similar in age.

Scale reliabilities were calculated to evaluate how well the items developed for each desirable property measured that property. The resulting scale reliabilities are presented in Table 2. Most of the scales attained minimum acceptable levels of scale reliability of .7 or above (Nunnally, 1978).

Desirable Property	**Cronbach's Alpha**	**# of Items**
Relative Advantage		
Life Cycle Applicability	.706	4
Normativeness	.786	3
Validity	.701	4
Predictiveness	Na	1
Timeliness	Na	1
Prescriptiveness	.859	4
Sensitivity	Na	1
Relative Advantage	**.891**	**18**
Compatibility		
Data Availability	.482	3
Language Independence	.877	3
Compatibility	**.699**	**6**
Complexity		
Automatable	.722	4
Calculation Ease	.751	3
Intuitiveness	Na	1
Understandability	.641	2
Complexity	**.804**	**10**

na – Not Applicable due to one item measure.

Table 2: Desirable Properties Scale Reliabilities.

The statistically significant comparisons of the means for the groups indicate that, overall, participants perceived FP to have greater relative advantage, compatibility, and complexity than SLOC. Results for software developers mirrored those of all participants; managers, however, did not perceive a significant difference on the relative advantage and compatibility constructs between FP and SLOC. Table 3 shows the associated means and p-values for the relative advantage, compatibility and complexity constructs.

Innovation Diffusion Construct	**All Participants means, p-value**	

Relative Advantage	FP = 3.69
	SLOC = 3.28, p=.002***	

Compatibility	FP = 3.65
	SLOC = 3.22, p=.015**	

Complexity*	FP = 2.93
	SLOC = 3.75, p=.000***	

	Developers means, p-value	**Managers Means, p-value**

Relative Advantage	FP = 3.79
	SLOC = 3.15, p=.001***
		FP = 3.61
		SLOC = 3.33, p=.145

Compatibility	FP = 3.83
	SLOC = 3.36, p=.055*
		FP = 3.49
		SLOC = 3.12, p=.166

Complexity*	FP = 2.90
	SLOC = 3.76, p=.000***
		FP = 2.95
		SLOC = 3.75, p=.000***

*** As the means for complexity increases, it indicates that the measure is decreasing in complexity**

***** p < .01, ** p < .05, * p < .10**

Table 3: Differences in Relative Advantage, Compatibility, and Complexity for all Participants, Developers, and Managers.

Developer and Manager Perceptions of FP and SLOC

Table 4 illustrates developer and manager perceptions of FP and SLOC.

Developers indicated that FP were significantly better than SLOC for all desirable properties of the relative advantage construct except sensitivity. This reveals that developers view FP as more applicable across the SDLC, have more normative standards, are better for predicting success, are more prescriptive, and provide more timely information than SLOC. Although managers and developers differed on several of the properties associated with relative advantage, they both agreed that FP are more valid than SLOC.

We also compared perceptions of desirable properties between managers and developers for FP and SLOC for the relative advantage construct (see column 3 of Table 4). There were no significant differences for any of the properties for SLOC, suggesting that both groups have similar views of the relative advantage of SLOC. Developers and managers differed on some of the properties for FP; developers believed more strongly than managers that FP are predictive and applicable across the life cycle. Managers believed more strongly than developers that FP are sensitive to changes in the software.

In terms of the compatibility construct, both developers and managers indicated that FP are more programming language independent than SLOC. Managers indicated that the data required for SLOC calculations are more readily available than the data required for FP calculations. Comparing the perceptions between managers and developers for FP and SLOC for the compatibility construct shows that developers believed more strongly than managers that FP are language independent.

In terms of complexity, developers and managers agreed that SLOC are easer to understand, easier to calculate and easier to automate than FP. Although managers indicated that SLOC behaves more according to intuition than FP, developers indicated that SLOC and FP are equally intuitive. Comparing perceptions between managers and developers for FP and SLOC for the complexity construct indicates that developers believed more strongly than managers that FP are intuitive.

IDT Construct ***Desirable property***	**Developer**	**Manager**	**FP** **SLOC**
Relative Advantage			
Life Cycle Applicability	FP: 4.19 SLOC: 3.54, p = .002 ***	FP: 3.83 SLOC: 3.12, p = .165	p = .038 ** p = .200
Normativeness	FP: 3.79 SLOC: 3.00, p = .006 ***	FP: 3.71 SLOC: 3.0, p = .240	p = .701 p = .255
Predictiveness	FP: 4.05 SLOC: 3.14, p = .038 **	FP: 3.44 SLOC: 3.47, p = .931	p = .017 ** p = .518
Prescriptiveness	FP: 3.42 SLOC: 2.86, p = .073 *	FP: 3.09 SLOC: 2.85, p = .465	p = .224 p = .991
Sensitivity	FP: 2.86 SLOC: 3.29, p = .353	FP: 3.64 SLOC: 3.53, p = .713	p = .012 ** p = .572
Timeliness	FP: 3.86 SLOC: 2.86, p = .004 ***	FP: 3.52 SLOC: 3.18, p = .328	p = .202 p = .514
Validity	FP: 3.95 SLOC: 3.21, p = .002 ***	FP: 3.92 SLOC: 3.54, p = .033 **	p = .780 p = .168
Compatibility			
Data Availability	FP: 3.48 SLOC: 3.52, p = .864	FP: 3.31 SLOC: 3.76, p = .038 **	p = .416 p = .360
Language Independence	FP: 4.17 SLOC: 3.19, p = .007 ***	FP: 3.68 SLOC: 2.57, p = .002 ***	p = .075 * p = .150
Complexity			
Automatibility	FP: 2.64 SLOC: 3.82, p = .000 ***	FP: 2.95 SLOC: 3.87, p = .000 ***	p = .168 p = .773
Calculation Ease	FP: 3.19 SLOC: 3.81, p = .004 **	FP: 3.12 SLOC: 3.73, P = .017 **	p = .435 p = .918
Intuitiveness	FP: 3.29 SLOC: 3.29, p = 1.00	FP: 2.60 SLOC: 3.12, p = .089 *	p = .020 ** p = .676
Understandability	FP: 2.81 SLOC: 3.79, p = .000 ***	FP: 2.86 SLOC: 3.85, p = .000 ***	p = .820 p = .825

***** p < .01, ** p < .05, * p < .10**

Table 4: Comparisons of Developer and Manager Perceptions of FP and SLOC for Desirable Properties.

DISCUSSION

Supported findings

For the relative advantage construct and its associated desirable properties, the perceptions of developers appear more congruent with the existing literature than the perceptions of managers. Developers indicated that FP have greater relative advantage than SLOC; managers, however, showed no significant differences for this construct. Developers indicated that FP are more applicable across the life cycle, more normative, more predictive, more prescriptive, and more timely than SLOC. Results for managers failed to achieve significance for any of these properties.

Although the literature appears inconclusive regarding whether FP are more compatible than SLOC, developers indicated that they view FP are more compatible than SLOC. Managers, however, did not perceive a significant difference in compatibility between FP and SLOC.

Prior research maintains that using FP are more complex than SLOC; results support this assertion for both managers and developers. Consistent with existing research, managers and developers agreed that the calculations for SLOC are easier to automate and that SLOC are easier to calculate. Managers also indicated that SLOC are more intuitive than FP.

Overall many of the findings and assertions of previous research were supported by the participants. This implies that practitioners are receiving the important messages associated with improving software development processes using software measures.

Unsupported findings

Contrary to assertions from prior literature, developers did not perceive significant differences between FP and SLOC for either the data availability or the intuitiveness properties. The literature suggests both that FP and SLOC exhibit validation issues; managers and developers, however, agreed that FP are more valid than SLOC. Additionally, although the prior literature appears inconclusive as to whether FP or SLOC are more understandable, both managers and developers agreed that SLOC are more understandable than FP.

Developer and Manager Perception Gap

Results indicate that developers and managers have different perceptions about FP and SLOC. Overall, developers' perceptions seem to be more congruent with the literature; that is, developers, more than managers, generally perceive the value of FP over SLOC. For example, developers perceive the relative advantage and compatibility of FP over SLOC, but it appears that managers do not. Furthermore, developers believed more strongly than managers that FP are predictive, language independent, intuitive, and applicable across the life cycle. These results suggest developers better recognize the benefits of FP.

Implications for practice

The findings in this study also help to explain why practitioners still use SLOC despite the advantages of FP. Both managers and developers agreed that SLOC are less complex than FP. Complexity negatively impacts the adoption of a software measure; thus, it seems that practitioners may still be using SLOC due to its simplicity. This implies that organizations should invest in training that attempts to reduce the perceived complexity of FP and highlights the greater relative advantage and compatibility advantages of FP over SLOC. Given that participants perceived greater complexity for FP than SLOC, future research should also investigate ways to reduce the complexity of FP calculations.

Perhaps more importantly, developers perceived a greater relative advantage and compatibility of FP over SLOC, but managers did not perceive significant differences for these constructs. This finding offers valuable insight into why practitioners continue to use SLOC. If managers fail to perceive the relative advantage and compatibility of FP over SLOC and perceive FP as more complex than SLOC, they may choose to adopt SLOC over FP. Managers, rather than developers, often make the decisions in information systems departments. Thus, managers may make the department-wide decision to use SLOC rather than FP because they fail to perceive the relative advantage and compatibility of FP and because SLOC are less complex than FP. Future research should investigate why managers do not perceive significantly greater relative advantage and compatibility for FP over SLOC.

Future research could investigate common IDT propositions relating the primary constructs to measures of adoption for developers and managers. Similarly, the inclusion of non-adopters with knowledge of software measures is necessary to determine if, as seems likely from the current study, that practitioners who select FP and practitioners who select SLOC have different causal models for the adoption of these software measures. Specifically, it seems that FP adopters may be more focused on relative advantage, while SLOC adopters are more focused on complexity. It may also be interesting to understand more about perceptions and goals of SLOC adopters through interviews or other qualitative techniques. Furthermore, future research could investigate possible causal relationships among the desirable properties. For example, life cycle applicability and language independence, even though assigned to different IDT constructs, address similar issues and therefore could exhibit interaction.

Limitations

The primary limitation with this study is small sample size; a total of 70 subjects answered the survey for FP and SLOC. Of the 70 subjects, 28 were developers and 42 were managers. Statistical power for the comparisons in Table 3 ranged from .45 to 1.0, averaging .85 for a significance level of .05. Thus, power of the sample is adequate.

A second limitation involves scale reliabilities. The scales for the data availability and understandability desirable properties are below the traditional .7 alpha cut-off. Initially, scales were developed for each desirable property. However, during the scale evaluation and refinement process, several items were eliminated, thereby resulting in some properties with single item measures.

CONCLUSION

While the literature has maintained that FP are the superior measure for sizing and productivity monitoring activities, researchers have not previously investigated practitioner perceptions of software measures using the theoretical framework of Innovation Diffusion. This study contributes to the literature by evaluating practitioner perceptions of FP and SLOC along the innovation diffusion dimensions of relative advantage, compatibility, and complexity. The findings in this study provide insight into why practitioners still use SLOC despite the benefits of FP and reveal a perception gap between developers and managers on several aspects of FP and SLOC.

Future research could investigate additional software measures in the context of innovation diffusion theory constructs and attempt to identify the relative influence of the desirable properties identified in this study. Ways to reduce the disparities in the views of managers and developers should result in greater adoption of FP, specifically education efforts directed at managers seems necessary.

REFERENCES

1. Albrecht, A. and Gaffney, J. (1983) Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Transactions on Software Engineering*, SE9, 6, 639-648.
2. Armour, P. (2004) Beware of Counting Code, *Communications of the ACM*, 47, 3, 21-24.
3. DeMarco, T. (1982) Controlling Software Projects, Yourdon Press, New York.
4. Dolado, J.J. (1997) A study of the relationships among Albrecht and Mark II Function Points, lines of code 4GL and effort, *The Journal of Systems and Software*, 37, 2, 161-173.
5. Fenton, N.E and Neil, M. (1999) Software metrics: Successes, failures and new directions, *The Journal of Systems and Software*, 47, 2, 149-57.
6. Fenton, N.E. and Pfleeger, S.L. (1996) Software Metrics: A Rigorous and Practical Approach 2nd edition, International Thomson Computer Press, London.
7. Grupe, F.H. and Clevenger, D.F. (1991) Using Function Point Analysis as a Software Development Tool, *Journal of Systems Management*, 42, 12, 23-26.
8. Hall, T. and Fenton, N. (1997) Implementing Effective Software Metrics Programs, IEEE Software, 14, 2, 55-64.
9. Heiat, A. and Heiat, N. (1997) A model for estimating efforts required for developing small-scale business applications, *The Journal of Systems and Software*, 39, 1, 7-14.
10. Henderson-Sellers, B. (1996) Object-Oriented Metrics: Measures of Complexity, Prentice-Hall, Upper Saddle, NJ.
11. Jones, C. (1991) Applied Software Measurement: Assuring Productivity and Quality, McGraw-Hill, New York.
12. Kemerer, C.F. (1987) An Empirical Validation of Software Cost Models, *Communications of the ACM*, 30, 5, 416-429.
13. Kemerer, C.F. and Porter, S. (1992) Improving the Reliability of Function Point Measurement: An Empirical Study, *IEEE Transactions on Software Engineering*, 18, 11, 1011-1024.

14. Kemerer, C.F. (1993) Reliability of function points measurement, *Communications of the ACM*, 36, 2, 85-97.
15. Low, G.C. and Jeffery, D.R. (1990) Function Points in the Estimation and Evaluation of the Software Process, *IEEE Transactions on Software Engineering*, 16, 1, 64-71.
16. Mahmood, M.A., Pettingell, K.J., and Shaskevich, A.I. (1996) Measuring productivity of software projects: A data envelopment analysis approach, *Decision Sciences*, 27, 1, 57-80.
17. Matson, J., Barret, B., and Mellichamp, J. (1994) Software Development Cost Estimation Using Function Points, *IEEE Transactions on Software Engineering*, 20, 4, 275-287.
18. Moser, S., Henderson-Sellers, B., and Misic, V. (1999) Cost estimation based on business models, *The Journal of Systems and Software*, 49, 1, 33-42.
19. Nunnally, J.C. (1978) Psychometric Theory 2nd edition, McGraw-Hill, New York.
20. Orr, G. and Reeves, T. (2000) Function Point Counting: One program's experience, *The Journal of Systems and Software*, 53, 3, 239-250.
21. Ray, G. (1993) The lines of code alternative, *Computerworld*, 27, 8, 91-92.
22. Rogers, E. (1983) Diffusions of Innovations 3rd edition, The Free Press, New York.
23. Tate, G. and Verner, J. M. (1991) Approaches to Measuring Size of Application Products with CASE Tools, *Information and Software Technology*, 33, 9, 622-628.