Educating Software Development Professionals: Does Instruction Affect Creativity?

Judy L. Wynekoop

Diane B. Walz

ABSTRACT: *Since creativity is important in software development, the effect of education in formal software development methods on individual creativity was studied. Students choosing to major in information systems and computer science in college were found to be more creative than the general population at the start of their programs. In fact, individuals choosing computer science as a major were exceptionally creative. However, by their senior year, computer science majors were no more creative than information systems majors. These findings support conclusions that computer science curricula should include more problem-solving and design activities. Implications for computer science and information systems curricula are discussed.*

KEYWORDS: *creativity, software development, education, information systems, computer science*

INTRODUCTION

Information systems are critical to corporate functioning, and businesses have increasingly complex information system needs. The production of high quality computer software and the efficiency of the software development process are key issues facing academics and practitioners today [1].

In an attempt to improve software quality and development productivity, standardized development methods and techniques have been created. The goals of these software engineering efforts include reduction of errors, faster development time, and reuse of code and design modules. To achieve these goals, software development methods and techniques often incorporate principles of standardization, reuse, and repetitiveness. While software productivity tools and formal methods offer some benefits in terms of faster development time and error management, these primarily benefit less complex applications and downstream activities like programming and testing [2].

The software design process is highly cognitive and intellectual, but not well-understood [3][4]. Recently, attention has been given to the increasing need for creativity in the development of complex and difficult software [5][6]. Within this context, fears have surfaced that the use of standard development methods and tools may suppress creative software solutions [7]. Deadlines for deliverables in traditional approaches to software design may offer disincentives for creativity in requirements analysis and solution design [6].

Although all individuals are creative to some extent, some individuals are intrinsically more creative than others. It is also known that creativity can be cultivated [8][9]. One study has shown that IS personnel exhibit different creativity styles than do individuals in other occupations and offers practical suggestions for adjusting managerial practices to the distinct characteristics of IS personnel [10]. Couger suggests techniques for stimulating creativity at specific points in the software development process [5] [6].

Little is known, however, about whether formal methods and standardized tools used for software development have any effect on the creativity of designers. The research reported here is a first step to determine whether the environment of software development is *nurturing*, *negative*, or *neutral* to the creativity of software developers.

In most cases, software developers are first exposed to standard methods, tools, and procedures in their undergraduate Computer Science (CS) or Information Systems (IS) degree programs. The purpose of this study is to determine whether this early formal education and training of potential developers has a nurturing, negative or neutral effect on their creativity.

We define the effect of the software development environment on creativity as *negative* if software developers are less creative after being exposed to the environment. This can be the result of constraints that make individuals less creative or that drive the more creative individuals out of the environment. The effect is *nurturing* if developers are more creative after being exposed to the environment. This would occur in an environment that stimulates individuals' creativity or that drives the less creative individuals away. The effect is *neutral* if there is no change in developers' creativity.

CS and IS programs are very different in orientation. Differences in the domains of IS and CS undergraduate education which might be considered to affect creativity are described below. Differences are identified using Amabile's [11] creativity framework. Based on the results of this analysis, we suggest two research questions about the effect of education on creativity. The results of the empirical analyses should prove useful both to those designing IS and CS curricula and to those managing the IS function in organizations.

CREATIVITY AND INFORMATION SYSTEMS DEVELOPMENT

The early stages of software development, analysis and design, are critical to the production of a quality software product. Different phases of software development involve different cognitive tasks [12]. It has been suggested that the early stages of software development require imagination and intuition [3][13]. Because the design process involves the formulation, refinement and simulation of solution models by cognitive activity, creativity is necessary in this problem solving process [14]. This contrasts with the later stages of software development (e.g. programming) which are more narrowly focused and precise [13].

The creativity of groups and individuals in an organization is shaped by the context created by the interaction of individuals, groups and the environment [15]. Creativity is not the same across all knowledge domains, but should be viewed as domain specific [11][16][17]. Individual level variables affecting individual creative performance include cognitive, personality, motivational and knowledge variables [15]. Amabile [11] provides the model of creative individual performance shown in Figure 1.

The model shown in Figure 1 contains three prerequisites for individual or group creativity: domain-relevant skills, creativity-relevant skills and task motivation. Domain-relevant skills include the knowledge and abilities needed to perform in a given area. Creativity-relevant skills are traits and abilities needed to invoke creative processes. Task motivation includes intrinsic and extrinsic motivational variables that will increase or decrease creative processes.

Some of the domain-relevant and creativity-relevant skills can be influenced by education and training. For instance, a person must have knowledge and skills in the domain of activity to be creative. In order to design an information system, an individual's knowledge must include analysis and design methods and techniques, knowledge of existing technologies, and knowledge of relevant functional areas. A designer's technical skills might include the ability to use technologies such as design software (e.g. CASE tools) and fourth generation languages. There is evidence that exposure to a wide variety of information in a domain enhances creativity [11]. Thus, there is some reason to believe that exposure through education to a variety of software design methods, techniques, and technologies might facilitate developer creativity.

Certain cognitive features are relevant to creativity in individuals. These include the ability to: understand complexities, see things differently from others, suspend judgment or commitment to a solution, see relationships between diverse pieces of information, recognize the importance of new information, and generate novel ideas [11] [14][18]. The ability to draw analogies and to combine knowledge from multiple areas is important in this endeavor [14][18]. Pure and independent ("bolt of lightning") insights are rare. Instead, the moment of insight represents the thought that integrates the concepts within the problem space, where everything falls into place [19].

Creative persons are able to break from the past and to identify exceptions and inconsistencies in the accepted way of doing things [18]. Johnson-Laird [20] describes constraints that are imposed by systems of rules within a domain and contends that there can be no real creativity without such constraints. In fact, this implies that creative persons are those who can rise above constraints. They can solve problems creatively within the confines of the domain.

Research indicates that expert designers exhibit these traits. The software design process involves the formulation, refinement and simulation of solution models by cognitive activity. Expert designers generate and evaluate more alternative solutions to subproblems, tackling the most complex ones first, drawing on a large experience-based store of solutions and solution methods [7][21][22]. The generation of many possible solutions and the application of solutions to different contexts is characteristic of creative people and is a skill that can be enhanced through training [5] [14][23].

Studies of large software development projects have identified exceptional designers who are considered to be essential to successful development projects [24]. These individuals can envision the interaction of various parts of the system and how it would behave, and can build new models to salvage failed projects. Expert software designers have also been found to mentally develop and simulate complex models of the software [3][7][25]. Such behavior is characteristic of creative people [9].

It appears that successful software designers exhibit traits associated with creative individuals. Although personality determines these traits to some extent, they also depend on experience and training [11]. It is not clear how training and experience in the field of software design affects individuals' creativity. It is possible that training in—and use of—formal design methods and techniques can help individuals develop domain- relevant and creativity-relevant skills. Or perhaps the formal tools and methods represent constraints that actually suppress creative software solutions.

Figure 1: Creativity Framework (Amabile, 1983)

Domain-Relevant Skills	Creativity-Relevant Skills	Task Motivation
Includes: • Domain factual knowledge • Required Technical skills • Domain relevant talent	Includes: • Appropriate cognitive style • Knowledge of heuristics for producing novel ideas • Conducive work style	Includes: • Attitudes toward task • Perceptions of own motivation for doing the task
Depends on: • Innate cognitive abilities • Innate perceptual & motor skills • Formal & informal education	Depends on: •Training • Experience in idea generation • Personality characteristics	Depends on: • Initial level of intrinsic motivation toward the task • Existing extrinsic constraints • Individual ability to cognitively minimize constraints

RESEARCH QUESTIONS

Training and creativity

It has been suggested that standard development methods might inhibit the creative processes which characterize outstanding designers [7][26][27]. Others have specifically recommended formal training in normative problem solving methodologies to enhance creativity [14].

Although formal education can increase creativity, too much formal education may decrease creativity by causing an individual to become too dependent on established algorithmic solutions to problems [11]. If so, then postsecondary programs in which students learn in this fashion may actually be "weeding out", or suppressing, creativity in individuals. Thus, the first research question addressed here is:

Does formal education and training in methods, techniques, and tools inhibit creativity in software designers?

This would be answered affirmatively if creativity decreases as students progress from lower division courses (i.e. freshman and sophomore) to upper division courses (i.e. junior and senior).

IS versus CS training

IS and CS curricula differ in fundamental ways. In IS programs, technical knowledge is taught within the context of organizations and management. CS programs, on the other hand, are rooted in mathematics, engineering, and algorithms. IS curricula emphasize problem solving methods and the process of application design and implementation of information systems within an organizational framework. CS majors typically receive less exposure to organizational considerations in developing information systems and requirements analysis, but more training in algorithm development, programming, hardware, and systems software [27][28].

Thus, the two curricula differ with respect to educational objectives which could have very different effects on creativity. From analysis of the specific content of both curricula, some differences and similarities were noted. Both IS and CS majors generally receive training in the areas of programming, database, and software development methods. A significant difference is noted, however, in the weights placed on the various topics and the overall focus of each curriculum.

The ACM-IEEE/CS Joint Curriculum Task Force [29] lists nine subject areas to comprise the CS discipline: algorithms and data structures, architecture, AI and robotics, database and information retrieval, human-computer communications, numerical and symbolic computation, operating systems, programming languages, and software engineering.

From current curriculum models and work in progress for the IS '95 Curriculum Model [30], subject areas that are relevant for the IS curriculum can be identified: design and implementation with database management systems, hardware and software, information systems theory, programming languages, project management, software engineering methods, systems analysis, systems design, and telecommunications.

For Information Systems, most AACSB accredited schools use the ACM (Association for Computing Machinery) IS curriculum model, which includes only two courses specifically focusing on programming while several courses include analysis and design [30][31]. These analysis and design courses include problem-solving skills and methods which have been shown to enhance creativity through techniques like identifying relationships between diverse pieces of information and diverging from the *status quo* [11][14].

Less than one-fifth of the recommended CS curriculum involves problem solving methods and concepts and the software development process [27]. The bulk of the curriculum involves programming languages and technical concepts, such as operating systems and hardware architecture. In fact, it is noted that "programming occurs in all nine subject areas" that are included in the curriculum [27, p. 77], and it is not clear how much the subject area including problem-solving will nurture creativity, since the subject is described as "...a rigorous introduction to the process of *algorithmic* problem solving...." [27, p. 83, italics added].

A solution is considered to be creative when it is both novel and relevant to the immediate task, and the solution to the task is not algorithmic and straightforward, but heuristic [11]. That is, discovering the problem is an important part of creativity. Thus, IS curricula appear to include more opportunities to teach creativity-relevant skills (Figure 1) than CS curricula, although both impart domain-relevant skills.

Most personnel enter the IS profession with an undergraduate degree in either IS (usually from a business school) or CS and begin their careers as programmers. However, IS majors generally follow a career path toward systems analyst or management positions, while CS majors tend to follow more technically focused careers [28].

Research has shown that individuals in traditional programming jobs (e.g. third generation languages) are left-brain dominant, exhibiting analytical and sequential thought, whereas fourth generation language programmers with a user orientation are more experimental, flexible and spontaneous [32]. These authors conclude that traditional computer science curricula train third generation language programmers.

IS graduates generally work in an organizational environment, interacting with both the organizational functional area environment and relevant information technologies. CS graduates tend to interact less with organizational functions and more with technology [27][28]. There is evidence that organizational computing departments continue to need both technically focused and business and end-user focused personnel, although the importance of the latter is expected to increase, while that of the former decreases [32][33].

"Although formal education can increase creativity, too much formal education may decrease creativity by causing an individual to become too dependent on established algorithmic solutions to problems"

IS majors seem to be exposed to more of the kind of problem solving activities which stimulate creativity than are CS majors [26][32]. Thus, the second research question in this study is:

Are information systems students more creative than computer science students?

RESEARCH METHODS

The California Psychological Inventory Adjective Check List (ACL) was used to measure creativity [34][35] of CS and IS students. The ACL is a widely used measure of adult creativity, showing high internal consistency

Table 1: Mean Creativity Scores

Major	Lower division			Upper division		
	Score	SD	n	Score	SD	n
IS	49.8	7.5	32	50.3	8.7	41
CS	54.2	7.2	28	50.5	7.8	53

Table 2: Results of two-tailed t-tests pairwise comparisons for differences in means

Comparison of means	IS t	CS t
Lower Division vs. Upper Division	-0.25	2.06*
(*denotes significance, α=.10)		

Table 3: Results of t-tests for pairwise comparison of IS & CS students

Comparison of means	t
All students	1.3
Upper division	0.1
Lower Division	2.3*
(*denotes significance, α=.10)	

reliability and good validity [9] [36][37]. The ACL is also not sensitive to training effects.

The ACL was administered to lower division (freshman or sophomore) and upper division (junior or senior students majoring in CS and IS. Lower division IS students were enrolled in the first programming course in the IS major; upper division students were enrolled in a required senior level IS projects course and an IS elective. Upper division IS students had completed required analysis and design and application development courses. Lower division CS majors were enrolled in the first programming course in the major; upper division CS majors were enrolled in two junior/senior level programming courses. Upper division CS majors had completed several programming courses, as well as courses in data structures and hardware architecture. The sample included 73 undergraduate IS majors (32 lower division and 41 upper division) and 81 undergraduate CS majors (28 lower division and 53 upper division).

In order to address the first research question, the ACL creativity scores of lower division students were compared to the scores of upper division students for both IS and CS majors. The IS and CS students were considered separately because the curricula are so different. If the educational process is suppressing creativity or driving out more creative individuals, the lower division students would be expected to have significantly higher scores than would the upper division students. For each comparison, two-tailed t-tests (α = .10) were performed to identify trends in creativity.

The second research question was addressed by comparing mean creativity scores of IS majors to the scores of CS majors, both in aggregate and by class (lower division and upper division). One-tailed t-tests (α= .10) were used to test the significance of the differences.

RESULTS

Mean creativity scores are shown in Table 1. It should be noted that the means scores of all the groups are above the ACL population norm of 48.5 for college students [35].

Table 2 presents the results of t-tests for differences in the means of lower division versus upper division students. For the IS undergraduate majors, there was no significant difference in mean creativity scores (t = -0.25, p = .81), indicating the design courses taken by the upper division students had not affected individual creativity. Both upper and lower division students had mean scores of approximately 50.

However, there were differences between the lower and upper division CS majors (t = 2.06, p = .04). Lower division students scored significantly higher than did upper division students, indicating either the curriculum negatively impacted creativity or that the more creative students left the major.

The second research question was addressed by comparing mean creativity scores of IS and CS undergraduate majors using one-tailed t-tests. The results are presented in Table 3. When lower and upper division students were combined, there was no significant difference in creativity scores between IS and CS students (t = 1.3, p = .18, DF = 152). However, when scores were compared within class levels, differences existed. However, the difference was not in the expected direction: creativity scores of lower division CS majors were significantly higher (t = 2.3, p = .03, DF = 58) than the scores of the lower division IS majors. The creativity scores of IS and CS upper division students were virtually identical (t= .14, p=.89, DF=92).

DISCUSSION

The data suggest that individuals preparing for careers in information systems through IS and CS undergraduate programs are more creative, on average, than the general population in the United States. These results can represent a benchmark for managers of IS professionals who are interested in providing creativity training for software developers.

The data also indicate that undergraduate CS programs attract more creative individuals than do IS programs. IS training does not appear to adversely affect individual creativity, and may actually improve it slightly. The CS program, on the other hand, did appear to have an adverse affect on creativity. Thus, it appears that CS programs are *negative* to software developers' creativity, while that conveyed by IS programs is at least *neutral* to creativity. From the empirical results we can infer that either creative individuals leave the major or that the training somehow stifles natural creativity. It has been suggested that computer science curricula over-emphasize left-brain thinking and should include more problem solving and design activities [26][32]. These findings support that suggestion.

Although third generation languages are still in use, and legacy systems will require maintenance in the future, the software development environment is changing. An increasing emphasis is placed on simultaneous thinking, flexibility, and client interaction [32]. Furthermore, the impact of these differences on the software product may be of concern. Amabile [11] has suggested that individuals with high initial domain-relevant skills and a low permanent repertory of cre-

ativity-relevant skills will produce products which are "predictable" but low in creativity. Individuals with high levels of both skills will produce creative products. Although creativity is important in software development [3][5][7], CS programs may be failing to convey creativity-relevant skills.

In conclusion, since creativity is important in software development, IS and CS curricula should incorporate methods which teach creativity-relevant skills (e.g. the ability to see patterns and relationships between diverse pieces of knowledge and the ability to break free from the past) in addition to the teaching of domain-relevant skills (e.g. design methods and programming languages). The focus of these methods should not be algorithmic problem-solving, but heuristic problem-solving, in which defining the problem and identifying a variety of procedures for solving it are part of the problem-solving process.

REFERENCES

[1] Pressman, R.S. (1992). *Software Engineering: A Practitioner's Approach*. 3rd edition. New York: McGraw-Hill.

[2] Forte, G. & Norman, R. (1992) CASE: A self assessment by the software engineering community, *Communications of the ACM*, 35 (4) 28-32.

[3] Guindon, R., Krasner, H., & Curtis, B. (1987). Breakdowns and processes during the early activities of software design by professionals. In Sheppard, S. & Soloway, E. (eds.) *Empirical Studies of Programmers Second Workshop*, 65-82, Norwood, NJ: Ablex.

[4] Kant, E. & Newell, A. (1985) Naive algorithm design techniques — a case study, in *Progress in Artificial Intelligence*, L. Steels and J. Campbell, eds., Chichester: Ellis Horwood Limited.

[5] Couger, J.D., Higgins, L.F., & McIntyre, S.C. (1993). (Un)Structured creativity in information systems organizations. *MIS Quarterly*, (17) 375-397.

[6] Couger, J.D. (1990). Ensuring creative approaches in information system design. *Managerial and Decision-Making Economics*, 11, 1268-1287.

[7] Glass, R.L. (1992). Creativity and software design: the missing link. *Information Systems Management*, 9(3), 38-41.

[8] Fontenot, N. (1993). Effects of training in creativity and creative problem finding upon business people. *The Journal of Social Psychology*, 133 (1), 11-22.

[9] Davis, G.A. (1992). *Creativity is Forever*. Dubuque: Kendall/Hunt.

[10] Miller, W.C, Couger, J.D. & Higgins, L.F. (1993). Comparing innovation styles of I.S. personnel to other occupations. *In Proceedings of the Hawaii International Conference on Systems Sciences* , pp. 378-386. .

[11] Amabile, T.M. (1983). *The Social Psychology of Creativity*. New York: Springer-Verlag.

[12] Laughery, Jr., K.R. & Laughery, Sr., K.R. (1985). Human factors in software engineering: A review of the literature. *The Journal of Systems and Software*, 5, 3-14.

[13] Dimino, S.A. & Nygren, C.M. (1985). There is a difference. *Journal of Systems Management*, 36(7), 34-36.

[14] Evans, J. (1993). Creativity in MS/OR: the multiple dimensions of creativity. *Interfaces*, 23(2), 80-83.

[15] Woodman, R.W., Sawyer, J.E., & Griffin, R.W. (1993). Toward a theory of organizational creativity. *Academy of Management Review*, 18 (2), 293-321.

[16] Li, J., & Gardner, H. (1993). How domains constrain creativity. *American Behavioral Scientist*, 37 (1), 94-101.

[17] Gardner, H. (1988). Creativity lives, creative works: A synthetic scientific approach. In R.J. Sternberg (ed.), *The Nature of Creativity*, New York: Cambridge University Press. 298-325.

[18] Bawden, D. (1986). Information systems and the stimulation of creativity. *Journal of Information Science*, 12(5), 203-216.

[19] Kneller, G.F. (1967) *The Art and Science of Creativity*, New York: Holt, Rinehart and Winston.

[20] Johnson-Laird, P.N. (1988) Freedom and constraints in creativity. In R.J. Sternberg (ed.), *The Nature of Creativity*, New York: Cambridge University Press. 202-219.

[21] Turner, J.A. (1987). Understanding the elements of system design. In Boland, R.J. & Hirschheim, R.A. (eds.). *Critical Issues in Information Systems Research*, 97-111, Chichester: John Wiley.

[22] Jeffries, R., Turner, A., Polson, P. & Atwood, M. (1981). The process involved in designing software, in *Cognitive skills and their acquisition*, J. Anderson, ed., Hillsdale, NJ: Lawrence Erlbaum Associates

[23] Guilford, J.P. (1967). Creativity: Yesterday, today and tomorrow. *Journal of Creative Behavior*, 1(1).

[24] Curtis, B., Krasner, H. & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, 31 (11), 1268-1287.

[25] Adelson, B. & Soloway, E. (1985). The role of domain experience in software design. *IEEE Transactions on Software Engineering*, SE-11(11), 1351-1360.

[26] Chesson, D., McBride, B. & Cartwright, C. (1992). Toward creativity: educating the college student in computer programming. *Journal of Research on Computing Education*, 25(2), 265-273.

[27] Turner, J.A. (1991). A summary of the ACM/IEEE-CS joint curriculum task force report: computing curricula 1991. *Communications of the ACM*, 34(6), 69-84

[28] Nunamaker, Jr., J.F., Couger, J.D. & Davis, G.B. (1982). Information systems curriculum recommendations for the 80s: undergraduate and graduate programs. *Communications of the ACM*, 25(11), 781-805.

[29] ACM/IEEE-CS Joint Curriculum Task Force. (1991) *Computing Curricula* 1991. February.

[30] Gorgone, J.T., Couger, J.D., Davis, G., Feinstein, D., Kasper, G. & Longenecker, Jr., H.E. (1994). Information systems '95 curriculum model — a collaborative effort. *Data Base*, 25(4), 5-8.

[31] Chen, J., Danesh, N.A. & Willhardt, J.A. (1991-1992). Computer curricula in AACSB-accredited business schools. *Interfaces*, 13(4), 60-72.

[32] Kettler, K., Smith, R.D. & Weinroth, J. (1992) Recruiting fourth-generation programmers. *Information Systems Management*, 9, 64-67.

[33] Jackson, D.P. (1991-1992). Curriculum design and the marketplace for MIS professionals. *Interface*, 13(4), 2-7.

[34] Gough, H.G. (1952). *Adjective Check List*. Palo Alto: Consulting Psychologists' Press.

[35] Gough, H.G. & Heilbrun, A.B. (1983). *The Adjective Check List Manual*. Palo Alto: Consulting Psychologists' Press.

[36] Davis, G.A. & Bull, K.S. (1978). Strengthening affective components of creativity in a college course. *Journal of Educational Psychology*, 70, 833-836.

[37] Domino, G. (1970), Identification of potentially creative persons from the Adjective Check List. *Journal of Consulting and Clinical Psychology*. 35, 48-51.

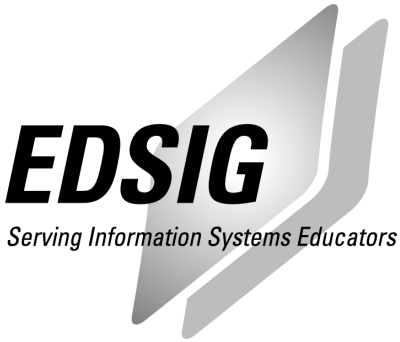
Judy L. Wynekoop
Diane B. Walz
Division of Accounting and Information Systems
University of Texas at San Antonio
San Antonio TX 78249-0632
jwynekoo@lonestar.jpl.utsa.edu

Judy L. Wynekoop is an assistant professor of information systems at the University of Texas at San Antonio. Her research interests include systems development and the use and impact of information technology in medicine.

Diane B. Walz is an associate professor of information systems at the University of Texas at San Antonio. Her research interests include personality dimensions of software designers and software tools for managing the principal-agent problem.

This research was partially funded by a UTSA Faculty Research Award.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096