The Two Cultures in Computing

Fred G. Harold
Department of Computer and Information Systems
Florida Atlantic University

Background

In 1956 the distinguished British scientist, novelist, essayist, and statesman C. P. Snow published a three-page observation titled "The Two Cultures" [Snow 1956], describing the distressing lack of communication between two of society's most gifted groups: scientific and literary intellectuals. This brief essay stimulated considerable comment, which led Snow to publish subsequent discussions of the same general theme [Snow 1959; Snow 1963].

The computing industry was barely established in 1956; in the ensuing thirty years it has become a major force in society. It has, in fact, generated many commentaries on the unique nature of the professionals engaged in its many dimensions. Specialties within computing today are as diverse as those in the medical profession.

Despite the many varieties of concentration within computing, two are most commonly represented in academic curricula (and in the ranks of practitioners as well): computer scientists and information systems specialists. Whether these professionals are designated systems programmers and systems analysts or software engineers and applications programmers, there is a significant distinction between their backgrounds, outlooks, and temperaments. In some instances the separation between these two groups is minimal; in others, it is considerable, and perhaps growing. There is justification (with apologies to the late Lord Snow (1905-1980) for adopting and somewhat modifying his model) for suggesting that two cultures exist even within the circumscribed arena of computing, and that the differing orientations of these two groups cause real problems for the larger profession.

The establishment of a profession requires, among other things, the acceptance of a common body of knowledge. This exists, although in fledgling fashion, in computer science (CS) today. The Computing Sciences Accreditation Board (CSAB) is in place, and as of June 1987 had accredited 48 university curricula in the U.S. Draft standards for the accreditation of information systems (IS) curricula have been developed. Leading professional societies, including ACM, DPMA, and the IEEE Computer Society, have participated in the recommendation of criteria for academic programs in one or both of these areas. CS emphasis is placed upon theoretical and mathematical foundations of computing, while IS concentration focuses on pragmatic applications and business uses of computers. Each of these sub-disciplines has little use for the other. Computer scientists denigrate the inelegant nature of the typical COBOL or spreadsheet business application, while information systems specialists decry the inability of the computer scientist to communicate effectively with users. The polarization between these two groups varies, but each typically views the other with suspicion and distrust.

This paper discusses the differing characteristics of these two groups, highlights the research which has identified these traits, and proposes a model for reintegration of their concerns under the heading of software engineering. C. P. Snow's work elaborating on his initial description of the Two Cultures is used as a framework for the discussion. The paper concludes with guidelines for the management of these two groups of professionals.

Characterization

Lest there be any misunderstanding, let me concede my own strong identification with the IS group rather than the CS contingent. I hope, however, that my membership in the IS subculture is not one which necessarily implies tunnel vision. I am quite aware that IS practitioners would do well to view CS professionals as role models in selected areas. In other dimensions of professional life, the reverse is true. Regardless of intended objectivity, I anticipate criticism--probably more vocal from CS than from IS "types".

It's probably a dangerous oversimplification to call CS adherents theorists and IS advocates practitioners, but as long as we remember the importance of both areas, as highlighted by the following paraphrased words of adult education specialist Dugan Laird, we can make use of the above terms:

> Theory without practice is mere speculation, while practice without theory is only anecdote.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1988 ACM 0-89791-262-4/88/0400-0188 $1.50

Several researchers have addressed the somewhat unique characteristics of computer professionals; perhaps the work of Couger and Zawacki [1978; 1981], Weinberg [1971], Shneiderman [1980], Perry and Cannon [1968], Cross [1970; 1971], Willoughby [1970], and Buie [1985] has been most productive. From the contributions of these researchers, the following characteristics of computer specialists (primarily programmers) have been generally accepted:

- low need for social interaction
- high need for achievement/challenge/growth
- low motivation towards management responsibilities
- low identification with authority
- low tolerance for interpersonal conflict
- loyalty to profession rather than employer
- optimism (especially regarding time estimates)
- preference for an unstructured (non-routine) environment
- systematic-methodical approach to problem-solving
- interest in stable, secure work
- high mechanical interest
- introverted, intuitive, thinking temperament--as opposed to extraverted, sensing, feeling orientation (measured by the MBTI, Myers-Briggs Type Indicator)

The problem with the above characteristics is that they have typically been applied to programmers and other computer specialists without regard for the particular orientation of their computing endeavors. It is the premise of this paper that very different temperamental profiles can be developed for CS and IS specialists--for theorists and practitioners. This premise is supported not by the results of experimental or survey research studies, but by observation over three decades of experience in a variety of business, government, and academic enterprises involving daily contact with both theorists and practitioners.

Distinctions

One of the most striking distinctions between these two groups is in their general political/social orientation. This difference, of course, is not absolute; rather it should be considered a statistical profile. Theorists tend to be liberal, both politically and socially, while practitioners fit a more conservative profile (perhaps assuming the characteristics of the business people they support). This contrast appears even in mode of dress, for (where such latitude is permitted, and even in some environments where it's frowned upon) the theorist tends to dress very informally, while the practitioner adopts a somewhat relaxed version of the business "uniform".

The theorist is interested in solving technical, typically abstract problems, appearing only marginally concerned with the practical application of such solutions; the practitioner, conversely, is committed to addressing very specific operational situations, sometimes failing to consider and adapt the theoretical contributions of the CS specialist.

The optimism mentioned in the above list is more a hallmark of the theorist than the practitioner. In their overriding concern for discovering the elemental algorithm for unraveling a frequently obscure problem, CS specialists often overlook mundane but very real hurdles that have moderated the expectations of their less sophisticated, and sometimes more jaded IS counterparts. The theorist has faith that "with one more little change, it's bound to work", while the practitioner knows better, based on harsh experience with a variety of inelegant applications. In an era of fascination with expert, knowledge-based systems, we still find glitches in the payroll program.

Perhaps the rifle and shotgun image can be applied here: The theorist concentrates on a constrained area, seeking to master its foundations and its abstract essence, while the practitioner ranges more widely, dabbling in many loosely coupled topics without a unifying focus.

Finally, the theorist, however socially and politically liberal, still remains something of an elitist, frequently expressing disdain for the routine drudgery of the practitioner. There is little doubt that the mathematical and abstraction capabilities of the average CS specialist exceed those of the representative IS professional; yet the ability of the latter to communicate with the layperson frequently outshines that of the theorist (although neither has cause for pride). The CS specialist is little concerned about this situation, preferring to seek the company of others with similar inclinations rather than to improve understanding within the larger computing community.

Although both subcultures reflect the attributes listed earlier, the theorist typically tends to display them with more intensity. The practitioner, however, lacks the unifying focus of the theorist. Each group can benefit greatly from the other; it is unfortunate that they usually show low tolerance for their opposite numbers. As Snow has said,

> Between the two a gulf of mutual incomprehension--sometimes (particularly among the young) hostility and dislike, but most of all lack of understanding. They have a curious distorted image of each other. Their attitudes are so different that, even on the level of emotion, they can't find much common ground. [Snow 1959, p. 15]

Synthesis

How can these two groups be reconciled? The following statement, although referring to a subset of the CS/IS dichotomy, can be extended to the larger context of the split between these two cultures: "We believe that both approaches have merit, and that all students in any type of computer science or information systems curriculum need to be exposed to the material of both" [Austing and Cassel 1988, p. xiv]. The answer is to be found in the discipline of software engineering. Since its inception in the late 1960's, software engineering has represented a unique blend of computing's two cultures: It has encompassed a broad range of concerns--from formal proofs of correctness to the development of clear specifications. It can serve to bring each of the groups closer to a balance from their positions at the theoretical and applied extremes of the computing continuum. The following statement summarizes this synthesis as it was intended when the term was originated:

> The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering [Buxton et al. 1976, p. 5].

The first issue of IEEE Transactions on Software Engineering identified the following as areas of software engineering concern:

- programming methodology
- software reliability
- performance and design evaluations
- software project management
- program development tools and standards [Yeh 1975, p. 1]

Engineers are in many ways the liaison between the world of pure or theoretical science and that of its pragmatic application. Software engineers can likewise link the world of theoretical computer science with the realm of applied information systems. The concept of "optimal heterophily" maintains that two individuals or groups can communicate most productively (if not always most comfortably) when they neither share too much in common nor have backgrounds that are too divergent: in short, when each has something to learn from the other but that "something" can be transmitted without excessive interference from cultural or ideological conflict. The arena of software engineering should serve as a "demilitarized zone" between the adversarial roles of CS theorists and IS practitioners. According to Fairley,

> Software engineering is a pragmatic discipline that relies on computer science to provide scientific foundations in the same way that traditional engineering disciplines such as electrical engineering and chemical engineering rely on physics and chemistry [Fairley 1985, p. 2; emphasis added].

There is a continuing discussion of the differentiation between systems analysis and design and software engineering; although the following is an oversimplification, the primary distinction seems to be in the amount of emphasis on the creation of the software itself at the code level--a function of software engineering more clearly than of systems analysis and design. Software engineering has traditionally been considered more a technological/computer science discipline than has the "softer" systems analysis and design endeavor; but in recent years the two have converged significantly. The result of this reintegration has been the emergence of software engineering as the most appropriate focus for research, education, and application of concepts in both information systems and computer science.

Conclusions

Given the breach between the Two Cultures in computing, how can we provide guidance for the effective management of these two diverse yet related groups in a manner that respects their separate identities and yet attempts to bring them closer? Not surprisingly, an important component of such a management agenda is cross-fertilization. Seminars and workshops dealing with "Concepts of A for B Specialists," where A and B can be information systems and computer science (or vice versa), will be difficult to sell at times, but valuable contributions to each discipline. University curricula in both specialties should include an overlap of several required courses (i.e., operating systems, computer organization, file and database structures, and data communications). Additionally, a software engineering course should be offered, team-taught by one instructor from each of the two disciplines.

Cross-fertilization in the workplace may be more difficult. Systems development teams can certainly be composed of representatives from each persuasion, but assignments in pure software creation, particularly one-person projects, will have to be undertaken by individuals with a strong predisposition to one of the two extremes.

Chauvinism of any form is regrettable; it is particularly insidious within a discipline whose professionals of differing outlooks must work in close harmony. Jokes and stereotypical comments are to be expected, but a gulf such as that described by Snow as the Two Cultures must be bridged. Orientation to the full complement of concerns addressed by software engineering is an important objective for any computing specialist, CS or IS in outlook, who seeks the breadth of the true professional without sacrificing depth in the chosen specialty.

Bibliography

Austing, Richard H. and Cassel, Lillian N. File Organization and Access. Lexington, Mass.: D. C. Heath and Company, 1988.

Buie, Elizabeth A. "Jungian Psychological Type and Programmer Team Building." Proceedings of IEEE COMPSAC. IEEE, 1985.

Buxton, John; Naur, Peter; and Randell, Brian (editors). Software Engineering: Concepts and Techniques. New York: Petrocelli/Charter, Inc., 1976.

Couger, J. Daniel and Zawacki, Robert A. Motivating and Managing Computer Personnel. New York: John Wiley and Sons, Inc., 1980.

________ "What Motivates DP Professionals." Datamation (September 1978), pp. 116-123.

Cross, Edward M. "The Behavioral Styles of Computer Programmers." Proceedings of the Eighth Annual Computer Personnel Research Conference. New York: Association for Computing Machinery, 1970, pp. 69-91.

________. "Behavioral Styles of Computer Programmers--Revisited." Proceedings of the Ninth Annual Computer Personnel Research Conference. New York: Association for Computing Machinery, 1971, pp. 140-167.

Fairley, Richard E. Software Engineering Concepts. New York: McGraw-Hill Book Company, 1985.

Perry, D. K. and Cannon, W. K. "Vocational Interests of Computer Programmers." Journal of Applied Psychology 52 (1968): 31-35.

Shneiderman, Ben. Software Psychology. Cambridge, Mass.: Winthrop Publishers, Inc., 1980.

Snow, Charles Percy. "The Two Cultures." New Statesman and Nation LIII (October 6, 1956): 413-414.

________. "The Two Cultures and the Scientific Revolution." (1959). Reprinted in Snow, C. P. Public Affairs. New York: Charles Scribner's Sons, Inc., 1971, pp. 13-46.

________. "The Two Cultures: A Second Look." (1963). Reprinted in Snow, C. P. Public Affairs. New York: Charles Scribner's Sons, Inc., pp. 47-79.

Weinberg, Gerald M. The Psychology of Computer Programming. New York: Van Nostrand Reinhold, Inc., 1971.

Willoughby, Theodore C. "Needs, Interests, and Reinforcer Preferences of Data Processing Personnel." Proceedings of the Eighth Annual Computer Personnel Research Conference. New York: Association for Computing Machinery, 1970, pp. 119-143.

Yeh, Raymond T. "Editor's Notice." IEEE Transactions on Software Engineering SE-1 1 (March 1975), pp. 1-6.