International Journal of Information Technologies and the Systems Approach

January-June 2008, Vol. 1, No. 1

# Table of Contents

## Editorial Preface



## Research Articles

# Information Systems, Software Engineering, and Systems Thinking:
## Challenges and Opportunities

*Doncho Petkov, Eastern Connecticut State University, USA*

*Denis Edgar-Nevill, Canterbury Christ Church University, UK*

*Raymond Madachy, University of Southern California, USA*

*Rory O'Connor, Dublin City University, Ireland*

## ABSTRACT

*This article traces past research on the application of the systems approach to information systems development within the disciplines of information systems and software engineering. Their origins historically are related to a number of areas, including general systems theory. While potential improvement of software development practices is linked by some leading experts to the application of more systemic methods, the current state of the practice in software engineering and information systems development shows this is some way from being achieved. The authors propose possible directions for future research and practical work on bringing together both fields with systems thinking.*

*Keywords:* *IS research; software development; software engineering; systems theory; systems thinking*

## INTRODUCTION

Information technology (IT) articles often include statements along these lines: "systems development continues to be challenging. Problems regarding the cost, timeliness, and quality of software products still exist" (Iivari & Huisman, 2007, p. 35). This recognition justifies the continuous search for improvement of Information Systems Development (ISD).

Glass, Ramesh, and Vessey (2004) provide an analysis of the topics covered by the three computing disciplines—information systems (IS), software engineering (SE), and computer science (CS)—and show overlaps between them all in the area of systems/software concepts. They also demonstrate that CS has only minor regard of the issues and concerns of systems/software management. Sommerville

(2007) states that CS is concerned with the theories and methods that underlie computers and software systems rather than the engineering and management activities associated with producing software. Whilst acknowledging that CS, SE, and IS do have a considerable overlap, the practices of both IS and SE have to deal with common matters such as the management of huge development projects, human factors (both software developers and software end users), organisational issues, and economic aspects of software systems development and deployment (Van Vilet, 2000).

For the reasons stated above, we will concentrate here only on SE and IS and their links to systems thinking. We will consider as a starting point the reality that the whole computing field has evolved historically as several "stovepipes of knowledge": CS, SE, and IS (Glass et al., 2004). Whether the separation or integration of computing disciplines will prevail is a complex issue. Integration has yet to be achieved as a consequence of the sets of values central to each area. We believe, along with others, that a systems approach may lead to improvement of the development and management of software systems and to a greater integration of computing. One might expect that the use of the word "system" in various contexts today leads to more "systems thinking," but is this true?

A reflective history of the IS field is presented in Hirschheim and Klein (2003, pp. 244-249). According to them, because of its roots in multiple disciplines, "such as computer science, management, and systems theory, it is hardly surprising that the field of IS cast a wide net when defining its boundaries, sweeping in many themes and boundaries" (Hirschheim & Klein, 2003, p. 245). In that light, it is somehow striking to note the conclusion about a lack of a systems approach in IS research according to Lee (2004, p. 16). Alter (2004) is even more specific, claiming that "the information systems discipline is ostensibly about systems, but many of our fundamental ideas and viewpoints are about tools, not systems" (p. 757).

The systems approach has been acknowledged in the SE literature as providing an insight into the factors that influence the success or failure of computer technologies (Mathieu, 2002, p. 138). It is symbolic that the 2006 special issue of the *IEEE Computer* magazine on the 60th anniversary of the IEEE Computer Society is dedicated to the past and future of software engineering. A brief examination of the papers in that issue shows that four of them are dealing with some systems features and the other three give examples of tool thinking. None of the seven papers in the issue had a reference to any source from the field of systems thinking and only one paper (Baresi, Di Nitto, & Ghezzi, 2006) had references to several classic SE sources dealing with fundamental systems ideas. This does not advance the ideas suggested by Boehm (2006a) and Sommerville (2007) that there is a need to integrate SE with systems engineering, a branch of systems thinking (see Jackson, 2003).

The contribution of this research is in the identification of areas where a systems approach would lead to improvements in ISD within a point of view that favors implicitly the integration of the IS and SE disciplines. The article will proceed with an analysis of how links between software development and systems thinking were perceived in the fields of IS and SE. This is done predominantly with the intention of exploring the application of systems ideas to software development separately in the two fields, outlining the success stories and the open problems. At the end, we will propose possible directions for future research in software development within SE and IS associated with the systems approach.

## ON INFORMATION SYSTEMS DEVELOPMENT AND SYSTEMS THINKING

A review of the history of various IS development methods is presented in Avison and Fitzgerald (2003). Iivari and Huisman (2007) point out, however, that the research literature on IS development has been scarce. This is most evident for the period after 1990. Prior to that point, the origins of IS research were associated

more strongly with issues on building information systems. However, one sub-area of IS development grew significantly in the U.K. and elsewhere over the last 20 years: incorporation of Soft Systems Thinking (SST) into IS.

## Soft Systems Thinking, Social Science, and Their Influence on IS

Stowell and West (1996) argued in the mid-1990s that practices of IS design had not appeared to have progressed since 1979; despite attempts in several proposals to embrace the social aspects of an information system, most seem to be based upon a functionalist view. Stowell and West (1996) explored the shift towards antipositivism in the mid-1980s, which resulted in a number of suggested methodologies that focused upon the social implications of computer systems design. As examples, they point out Soft Systems Methodology (SSM) (Checkland, 1999), the MULTIVIEW approach (Avison, 2000), participative systems design, and others (see also Avison & Fitgerald, 2003).

SSM evolved originally from experience within interventions in various management problems in public administration and industrial companies. However, subsequently it evolved more towards the field of IS (see Checkland & Holwell, 1998). Stowell (1995) presents a collection of papers analysing various aspects of the contribution of SSM to IS. SSM seems to be the most well researched interpretive systems approach used in the field of IS (see Holwell, 2000, for a detailed account of the literature on SSM, and Checkland & Poulter, 2006, for a contemporary presentation of SSM ideas).

The relevance of SSM to the field of IS has been explored in two directions. One way is to apply SSM on its own in some IT related aspect, for example, extend the standard SSM method to specify the information requirements of the system (see Wilson, 1990). The use of SSM in data modeling is explored by Lewis (1995). A further application of SSM for improvement of software quality is presented in Sweeney and Bustard (1997). A second direction of using SSM in information systems is through the linking of SSM to existing design methods. An overview and detailed analysis of using SSM with structured analysis and design is provided by Mingers (1995). Several authors have covered aspects of combining the Unified Modeling Language (UML) with SSM. A recent paper by Sewchuran and Petkov (2007) analyses the related theoretical issues and shows a practical implementation of a combination of UML and SSM within a Critical Systems Thinking (CST) (see Jackson, 2003) framework justified by Multimethodology (see Mingers, 2001).

## On Critical Systems Thinking, Multimethodology, and IS

Multimethodology is a metatheory for mixing methods from different methodologies and paradigms in the same intervention (Mingers, 2001). It seems to be an attractive vehicle for further research in systems thinking and IS research. Further refinement of the ideas on pluralist interventions can be found in a recent paper on Creative Holism (Jackson, 2006). Details on three cases, illustrating how Multimethodology and CST were practiced in separate systemic interventions in the Information and Communications Technologies sector, can be found in Petkov, Petkova, Andrew, and Nepal (2007).

In his paper on the links between CST and IS research, Jackson (1992) demonstrates the power of an integrated critical approach in the IS field. However, there have been relatively few subsequent publications on the practical application of CST in IS. Some of them are surveyed in Ngwenyama and Lee (1997), a paper demonstrating the significant relevance of CST to IS. Another interesting example, exploring how Triple Loop Learning (Flood & Romm, 1996) can be applied to the complexities during systems development is given in Finnegan, Galliers, and Powell's (2002) work. Further papers on systems thinking and IS can be found in proceedings of several meetings on the philosophical assumptions of IS research that took place after 1997, including the U.K. Annual Systems Conference, the European Conference on Information Systems, the Australasian

Conference on IS, and Americas Conference on Information Systems (AMCIS).

CST provides both theoretical sophistication and practical directions for future research that are applicable to IS. Jackson (2003) cautions that whatever argument is made in favour of pluralism, it is bound to run up against objections from those who believe in the incommensurability of paradigms. The latter notion is linked to the assumption that if paradigms have distinct and opposing philosophical foundations, applying them together is impossible. This issue has been addressed by several authors in the past (see Jackson, 2003). Zhu (2006), however, questioned recently the relevance of concerns about paradigm incommensurability from a practical point of view, another issue for possible further research. His view on paradigm incommensurability is similar to that of the pragmatic pluralism approach. This is based on the assumption that we are witnessing the end of a particular reading of theory and that there is no single truth and no single rationality (White & Taket, 1996, p. 54).

Both pragmatism and functionalism are often criticised in systems thinking (see Jackson, 2003). However, an interesting and relevant new systems approach in IS, the work system method (Alter, 2007), has emerged recently that may be linked to the pragmatic school of thought.

## The Work System Method and IS

Alter (2006) stresses that past dominance of single ideas like Total Quality Management and Business Process Re-engineering are not sufficient to influence the IS field profoundly. The work system method provides a rigorous but nontechnical approach to any manager or business professional to visualise and analyse systems related problems and opportunities (Alter, 2006). This method is more broadly applicable than techniques "designed to specify detailed software requirements and is designed to be more prescriptive and more powerful than domain-independent systems analysis methods such as soft system methodology" (Alter, 2002). We may note that making comparisons between the work system method and soft systems methodology requires a broader investigation of their philosophical assumptions and scope. A possible starting point for comparing their areas of applicability could be the classification of strategies for doing systems analysis provided by Bustard and Keenan (2005). SSM has been attributed by them to the situation when the focus is on development of a long term vision of the environment in which a computer system is to be used with identification of appropriate organisational changes (see Bustard & Keenan, 2005). Where Alter's approach stands in the Bustard and Keenan (2005) classification is an open question for research requiring both theoretical work and field experimentation. We consider the systemic nature of the work system method and its applicability to understanding business and IS problems to be its most distinctive and important characteristics. Though the work system method has a relatively short history and a small group of followers for now, the multifaceted scale of Alter's work, bringing together systems ideas with methods for deeper understanding of work systems and IS, has strong appeal.

## On Sticking to a Single Research Tradition in IS

Bennetts, Wood-Harper, and Mills (2000) provide an in-depth review of combinations of SSM with other IS development methods supporting multiple perspectives along the ideas of Linstone (1984). Thus, they brought together two distinct traditions in IS research: the former practiced in U.K./Europe/Australia where SSM has found significant acceptance, and the latter was pursued predominantly in the U.S. Linstone's ideas are strongly related to the influence of Churchman whose analysis of Inquiring Systems was a starting point for some significant IS research that followed (e.g., Vo, Paradice, & Courtney, 2001).

It is interesting to note that Bennetts et al. (2000) have examined sources not only from IS but also from the CS and SE literature. This raises a question that is hard to answer in a simple way. We observe that often authors of SE articles belong to CS or IS departments,

rather than engineering schools (Aurum & Wohlin, 2005; Dietrich, Floyd, & Klichewski, 2002). On the other hand, it seems that publications on IS development written by U.S. scholars often use references only from IS or from SE disciplines, depending on the field of the authors; a refreshing exception is a series of articles written over many years by R. Glass and I. Vessey with several collaborators (Glass et al., 2004). The reason could be the lack of communication between CS, SE, and IS (see Glass, 2005). Another possible reason is the growing concern within the separate computing fields for promoting and protecting their own paradigms (Bajaj, Batra, Hevner, Parsons, & Siau, 2005).

Maybe similar paradigmatic concerns have led Allen Lee to formulate his first idea from an advice to IS researchers: "practice paradigm, systems thinking and design science" (Lee, 2000). These are seen as a recipe to address the three dilemmas that are as relevant today as they were in 2000: the rigor vs. relevance debate in IS research; the "reference discipline" vs. "independent discipline" dilemma; and the technology vs. behaviour as a focus for IS research dilemma.

So far, we have considered the second of Lee's ideas and its relevance to IS development over the last 15 years and to a lesser degree some issues related to scientific paradigms in terms of Kuhn (1970). Further details on earlier contributions of Systems Science in the 1970s and 1980s can be found in comprehensive reviews related to the fields of IS research (see Xu, 2000), Decision Support Systems (see Eom, 2000), and Information Resources Management (see McLeod, 1995). Mora, Gelman, Forgionne, Petkov, and Cano (2007) presented a critique and integration of the main IS research paradigms and frameworks reported in the IS literature using a systems approach. We briefly comment below on design science, a more recent trend in IS research.

## On Design Science As One of the Directions to Resolve the Three Dilemmas in IS

According to Hevner, March, Park, and Ram (2004), IS related knowledge is acquired through work in behavioural science and design science paradigms. They point out that "behavioral science addresses research through the development and justification of theories that explain phenomena related to the identified business need, while design science addresses research through the building and evaluation of artifacts designed to meet the particular need." Another relevant detail is the differentiation that Hevner et al. (2004) make between routine design and system building from design science. The former is associated with application of existing knowledge to organisational problems, while the latter is associated with unique (often wicked or unresolved) problems that are associated with the generation of new knowledge. The latter idea is similar to the main thesis in Hughes and Wood-Harper (1999). Hevner et al. (2004) laid the foundation for a significant boost in IS research on issues related to IS development, including systems analysis and design science. The journal *Communications of AIS* started a series of articles in 2005 on this topic; the first of which was Bajaj et al. (2005). We may note that in spite of progress in applying action research in IS in theory (see Baskerville & Wood-Harper, 1998) and in practice (see the IbisSoft, n.d., position statement on environment that promotes IS research) the dominant IS research trend has been of a positivist behavioural science type which is another challenge for the proponents of a systems approach.

A substantial attempt to provide suggestions towards resolving the three dilemmas in IS research mentioned by Lee (2000) is discussed in Hirschheim and Klein (2003). They identify a number of disconnects between various aspects of IS research and outline a new body of knowledge in IS development (Iivari, Hirschheim, & Klein, 2004). They suggest there are five knowledge areas in ISD: technical knowledge, application domain (i.e., business function) knowledge, organisational knowl-

edge, application knowledge, and ISD process knowledge. Further, according to Hirschheim and Klein (2003):

*ISD process knowledge is broken down into four distinctive competencies that IS experts are suggested to possess: (1) aligning IT artefacts (IS applications and other software products) with the organizational and social context in which the artefacts are to be used, and with the needs of the people who are to use the system as identified through the process of (2) user requirements construction…(3) organizational implementation from which (4) the evaluation/assessment of these artefacts and related changes is factored out …. These competencies are ... at best weakly taken into account in the ten knowledge areas of SWEBOK.* (see for comparison SWEBOK, 2004)

Hirschheim and Klein (2003) present comprehensive proposals for strengthening the IS field. Their work was partly motivated by a widely discussed paper by Benbasat and Zmud (2003) on the identity crisis in the IS discipline. Both papers provide important background details about the IS research environment in which one may pursue the main ideas of this article. The next section will explore the relevance of systems thinking to SE.

## ON SOFTWARE ENGINEERING AND SYSTEMS THINKING

Software engineering has a primary focus on the production of a high quality technological product, rather than on achieving an organisational effect, however increasing emphasis in SE is being given to managerial and organisational issues associated with software development projects. Cornford and Smithson (1996) observe that SE "can never encompass the whole range of issues that need to be addressed when information systems are studies in the full richness of their operational and organisational setting".

Weinberg (1992) writes about systems thinking applied to SE. It is an excellent introduction to systems thinking and quality software management dealing with feedback control. It has a close kinship with the concepts of systems thinking and system dynamics in Madachy (2007), even though it is almost exclusively qualitative and heuristic. Weinberg's main ideas focus around management thinking about developing complex software systems, having the right "system model" about the project and its personnel.

Systems thinking in the context of SE, as described in Madachy (2007), is a conceptual framework with a body of knowledge and tools to identify wide-perspective interactions, feedback, and recurring structures. Instead of focusing on open-loop event-level explanations and assuming cause and effect are closely related in space and time, it recognises the world really consists of multiple closed-loop feedbacks, delays, and nonlinear effects.

Lee and Miller (2004), in their work on multiproject software engineering, advocate a systems thinking approach as "in general, we are able to make better, more robust, and wiser decisions with systems thinking, since we are considering the problem by understanding the full consequences of each feasible solution".

Other details on systems thinking with links to other books and articles can be found through practitioner's Web sites such Weinberg (2007), Developer (2007), or Yourdon (2007). The interest of software practitioners in systems ideas is a significant fact, in light of the previously mentioned debate about relevance in the IS literature. However, systems thinking is not mentioned by Reifer (2003) in his taxonomies of the SE theory state-of-the-art and SE state of practice. In relation to that, we will discuss below whether systems ideas are promoted in SE education.

### Software Engineering Education and Systems Thinking

The coverage of systems concepts in leading SE textbooks is possibly another indicator about the way the systems approach is perceived within the SE community. We considered books by several well established authors: Sommerville (2007), Pressman (2001), and Pfleeger (2001), amongst many. Table 1 shows a summary of

findings related to the treatment of several typical systems notions in those books.

Table 1 shows that the systems concepts covered in the three widely used textbooks are mostly related to introductory notions from systems thinking. There is nothing about open and closed systems, about the law of requisite variety or any other aspect of cybernetics, very little about sociotechnical systems, and nothing about soft systems methodology or CST. In our opinion, these are unexploited notions that have some potential to introduce fresh ideas in SE after further research.

Crnkovic, Land, and Sjogren (2003) question whether the current SE training is enough for software engineers. They call for making system thinking more explicit in SE courses. They claim that "the focus on modifiability (and on other non-functional properties) requires more of a holistic and system perspective" (Crnkovic et al., 2003). Similar thoughts are shared more recently by others in engineering like Laware, Davis, and Perusich (2006).

The narrow interpretation of computing disciplines is seen as a contributory factor to the drop in student enrolments in the last five years. Denning (2005) hopes that students will be attracted by a new educational approach promoted by the ACM Education Board that relies on four core practices: programming, systems thinking, modeling, and innovating. It has now been four years since those ideas were stressed by ACM, but there is little evidence that systems thinking has become a core practice emphasised in teaching in any of the three computing disciplines.

In the U.K., the Quality Assurance Agency (which monitors and quality assures all U.K. university programmes) recently published the updated version of the computing benchmark statement (encompassing IS, SE, and CS) on the content and form of undergraduate courses (QAA, 2007). Although not intended to be an exhaustive list but "provided as a set of knowledge areas indicative of the technical areas within computing," it fails to make explicit reference

*Table 1. Systems features covered in popular software engineering textbooks*

| **Notions covered** | **Author** | | |
|---|---|---|---|
| | Sommerville | Pressman | Pfleeger |
| System definition | Yes | Yes | Yes |
| Boundary | Implied | Yes | Yes |
| Open vs Closed systems | No | No | No |
| Relationships | Implied | Implied | Yes |
| Inter-related systems | Implied | Implied | Yes |
| Emergent property | Yes | No | No |
| Decomposition | Yes | Yes | Yes |
| Coupling | No | Yes | Yes |
| Cohesion | No | No | Yes |
| Hierarchy | Yes | Yes | Yes |
| System behaviour | Yes | Yes | Yes |
| Law of requisite variety | No | No | No |
| Sociotechnical systems | Yes | No | No |
| Systems engineering | Yes | To some extent | To some extent |

to systems thinking or systems approaches and makes only one reference to "systems theory" under a more general heading of "systems analysis and design". Perhaps the answer is to explore how to introduce these concepts earlier in pre-university education or to continue to try to convince the broader academic community of the importance of systems thinking.

One promising systems approach used for education of software engineers is the Model-Based System Architecting and Software Engineering (MBASE) framework being used at USC, and also adapted by some of their industrial affiliates. According to Boehm (2006c), MBASE integrates the systems engineering and SE disciplines, and considers stakeholder value in the system development. The MBASE framework embodies elements of agile processes and teaches students to "learn how to learn" as software development will continue to change. Valerdi and Madachy (2007) further describe the impact of MBASE in education.

## On Software Engineering and Systems Engineering

Systems Engineering is concerned with all aspects of the development and evolution of complex systems where software plays a major role. Systems engineering is therefore concerned with hardware development, policy and process design, and system deployment, as well as software engineering. System engineers are involved in specifying a system, defining its overall architecture, and then integrating the different parts to create the finished system. Systems engineering as a discipline is older than SE, as people have been involved in specifying and assembling complex industrial systems such as aircraft and chemical plants for more than 100 years (Sommerville, 2007).

A thought provoking comparison of SE culture vs. systems engineering culture is presented by Gonzales (2005). This work points out to where we should strive to change the perceptions of the SE student entering the IT profession. We agree with Gonzales (2005) that we "must continue the dialogue and ensure that we are aware of strides to formalise standard systems engineering approaches and generalise software engineering approaches to capturing, specifying and managing requirements" (p. 1). We would also suggest that this dialogue should be supported by more work on the application of a systems approach to SE, stimulated by journals such as IJITSA.

Boehm (2006b) concludes that "The push to integrate application-domain models and software-domain models in Model Driven Development reflects the trend in the 2000's toward integration of software and systems engineering". Another reason he identifies is that other surveys have shown that the majority of software project failures stem from systems engineering shortfalls. A similar thought is expressed by Boehm and Turner (2005), who state that there is a need to move towards a common set of life-cycle definitions and processes that incorporate both disciplines' needs and capitalise on their strengths.

Boehm (2006a) points out that "recent process guidelines and standards such as the Capability Maturity Model Integration (CMMI), ISO/IEC 12207 for software engineering, and ISO/IEC 15288 for systems engineering emphasise the need to integrate systems and software engineering processes". He further proposes a new process framework for integrating software and systems engineering for 21st century systems and improving the contractual acquisition processes. Another issue is how to capitalise on the new developments in SE over the last decade which will be discussed in the next section.

## The Evolution of Plan-driven and Agile Methods in SE and System Thinking

The traditional software development world, characterised by software engineering advocates, use plan-driven methods which rely heavily on explicit documented knowledge. Plan-driven methods use project planning documentation to provide broad-spectrum communications and rely on documented process plans and product plans to coordinate everyone (Boehm & Turner, 2004). The late 1990s saw

something of a backlash against what was seen as the over-rigidity contained within plan-driven models and culminated in the arrival of agile methodologies, which rely heavily on communication through tacit, interpersonal knowledge for their success.

Boehm and Turner (2004) quote Philippe Kruchten (formerly with IBM Canada and now a professor at UBC in Vancouver) who has likened the Capability Maturity Model (CMM)—a plan-drive approach—to a dictionary:

*'that is, one uses the words one needs to make the desired point; there is no need to use all the words available' (p. 23). They conclude that processes should have the right weight for the specific project, team, and environment. Boehm and Turner (2004) have produced the first multifaceted comparison of agile and plan-driven methods for software development. Their conclusions show that neither provides a 'silver bullet' (Brooks, 1987). Some balanced methods are emerging. We need both agility and discipline in software development.* (Boehm & Turner, 2004, p. 148)

Boehm (2006b) presents a deep analysis of the history of SE and of the trends that have emerged recently. These include the agile development methods: commercial off-the-shelf software and model driven development. The same author points out that the challenges are in capturing the evolving IT infrastructure and the domain restructuring that is going on in industry. In our opinion, it is necessary to investigate further if systems thinking may play a role in integrating agile and plan-driven methods (see Madachy, Boehm, & Lane, 2007, as an application of systems thinking to this problem). It has also been speculated that systems thinking could be relevant to Extreme Programming (XP) as it supports building relevant mental models (see Wendorff, 2002).

A recent paper by Kroes, Franssen, van de Poel, and Ottens (2006) deals with important issues in systems engineering such as how to separate a system from its environment or context. They conclude that the idea that a sociotechnical system can be designed, made, and controlled from some central view of the function of the system has to be given up, as many actors within the sociotechnical system are continuously changing (redesigning) the system. This is an important issue deserving further investigation in light of software systems and the methods implied by agile development frameworks.

## Systems Dynamics and SE

A widely publicised idea is modeling software development processes through systems dynamics (see Abdel-Hamid & Madnick, 1991; Madachy, 2007; and others). The differences and relationships between systems dynamics and systems thinking are detailed in Richmond (1994) and others. Systems dynamics is a tool that can assist managers to deal with systemic and dynamic properties of the project environment and can be used to investigate virtually any aspect of the software process at a macro or micro level. It is useful for modeling sociotechnical factors and their feedback on software projects. The systems dynamics paradigm is based on continuous systems modeling, which has a strong cybernetic thread. Cybernetic principles are relevant to many types of systems including software development systems, as detailed in Madachy (2007).

The primary purposes of using systems dynamics or other process modeling methods in SE as summarised from Madachy (2007) are strategic management, planning, control and operational management, process improvement and technology adoption, and training and learning. Example recent work by Madachy (2006) focuses on the use of systems dynamics to model the interaction between business value and the parameters of a software process for the purpose of its optimisation. Another application of systems dynamics to assess a hybrid plan-driven and agile process that aims to cope with the requirements of a rapidly changing software environment while assuring high dependability in Software-Intensive-Systems-of-Systems (SISOS) is presented in Madachy, Boehm, and Lane (2007).

### On Other Methods of Systems Thinking Applicable to SE

The development of understanding of a particular software project for making better judgments about the cost factors involved in cost and effort estimation is supported also by the work of Petkova and Roode (1999). They implemented a pluralist systemic framework for the evaluation of the factors affecting software development productivity within a particular organisational environment. It combines techniques from several paradigms: stakeholder identification and analysis (from SAST, see Mason & Mitroff, 1981), from SSM (Checkland, 1999), Critical Systems Heuristics (Ulrich, 1998), and the Analytic Hierarchy Process (Saaty, 1990).

While we could not find any specific earlier accounts of the use of SSM in the mainstream SE literature, it is significant that Boehm (2006a) has recognised its potential as he quotes its originator in a recent paper:

Software people were recognising that their sequential, reductionist processes were not conducive to producing user-satisfactory software, and were developing alternative SE processes (evolutionary, spiral, agile) involving more and more systems engineering activities. Concurrently, systems engineering people were coming to similar conclusions about their sequential, reductionist processes, and developing alternative "soft systems engineering" processes (e.g., Checkland, 1999), emphasising the continuous learning aspects of developing successful user-intensive systems.

One does not need always to have a systems philosophy in mind to generate an idea that has a systemic nature or attempts to change the current thinking in SE. Thus, Kruchten (2005) presents, under the banner of postmodernist software design, an intriguing framework for software design borrowed from architecture. One may investigate how such an approach is different from a systemic methodology and what are their common features. Starting from a language-action philosophy point of view, Denning and Dunham (2006) develop a framework of innovation based on seven practices that are inter-related in their innovation model—every element is in a relationship with all others, thus fulfilling the criterion for "systemicity" by Mitroff and Linstone (1993). We need more analogical examples of systemic reasoning or even just of alternative thinking related to every aspect of the work of a software engineer and IS developer demonstrating the power of innovative interconnected thinking. The analysis so far allows us now to formulate some recommendations in the following section.

## CONCLUDING RECOMMENDATIONS ON THE NEED FOR MORE RESEARCH LINKING SOFTWARE ENGINEERING, INFORMATION SYSTEMS DEVELOPMENT, AND SYSTEMS THINKING

We may derive a number of possible directions for future work from the analysis of research and practice in ISD and systems thinking within the fields of IS and SE. Alter (2004) has produced a set of recommendations for greater use of systems thinking in the IS discipline which incorporate various aspects of the work system method. We believe that Alter's proposals are viable and deserve the attention of IS and SE researchers.

Boehm and Turner's (2005) suggestions to address management challenges in integrating agile and plan-driven methods in software development will be used by us as an organising framework for formulating directions for research on integrating IS, SE, and the systems approach. The five main points below are as defined originally by Boehm and Turner (2005) for their purpose, while we have provided for each of them suggestions promoting such integration along the aims of this article:

1. **Understand how communication occurs within development teams:** There is a need to continue the work on *integrating systemic methods promoting organisational learning* (see Argyris & Schon, 1978) like systems dynamics, stakeholder analysis, soft systems methodology, critical

systems thinking, and others to identify the advantages of using specific methods and their limitations when dealing with uncovering the microclimate within a software development team. Most of the previously mentioned applications of systems methods for this purpose have had limited use and little experimental evaluation. *More case studies need to be conducted in different software development organisations to validate the claims for the applicability of such methods and to distil from the accumulated knowledge best practices and critical success factors* relevant to flexible, high quality software development teams. We may *expand further the boundary of investigations with respect to what is happening at the level of systems-of-systems* (see Sage, 2005). An example of related relevant ideas on cost estimation for large and complex software projects can be found in Lane and Boehm (2007). Another direction is to *explore information systems development as a research act*, as suggested by Hughes and Wood-Harper (1999) and Hevner et al. (2004), as well as the philosophy of integrating practice with research in the field of software and management, promoted by IbisSoft (n.d.).

2. **Educate stakeholders:** This is probably the most difficult task of all. It needs to be addressed at several levels:
   - *Implement changes in educational curricula*—it is essential to introduce the systems idea in relatively simple forms at the undergraduate level and in more sophisticated detail at the masters' level. There is *a need to create the intellectual infrastructure for more doctoral dissertation projects in IS or SE involving systems thinking*. Teaching could be supported by *creating an accessible repository for successful utilisation of systems ideas in IT education*. Amongst the many examples we may mention here the use of SSM in project-based education at a Japanese university (Chujo & Kijima, 2006), on integrating systems thinking into IS education (see Vo, Chae, & Olson, 2006), or the use of MBASE in student projects (see Boehm, 2006c; Valerdi & Madachy, 2007).
   - *Broaden the systems knowledge of IS and software engineering educators*—the current situation in some of the computing disciplines can be compared to a similar one in Operations Research (OR) in the 1960s, which had evoked a sharp critique by Ackoff (1999) in his famous paper "The Future of Operational Research Is Past." published originally in 1979. Ackoff (1999, p. 316) points that survival, stability, and respectability took precedence over development and innovativeness in OR in the mid-1960s and its decline began. The *challenge however is not just to bring systems thinking to IS and SE education beyond several elementary concepts of general systems theory but to keep up to date with the latest body of knowledge in the systems field*. For a comprehensive overview, see Jackson (2003) and, for recent developments in systems science, see Barton, Emery, Flood, Selsky, and Wolstenholm (2004).
   - *Empower IT developers to practice systemic thinking*—a significant role here needs to be played by research on the most *suitable forms for continuing professional education on IT and the systems approach, supported by professional meetings and journals for mixed audiences* like this one, that are oriented to academia and industry practice. Ackoff (2006) underlines that one of the reasons why systems ideas are adopted by few organisations is that "very little of the systems literature and lectures are addressed to potential users" (p. 707). Further, he stresses the need to analyse manage-

ment failures systemically, pointing out that there are two types of failures: errors of commission and errors of omission. In spite of publications analysing software failures like Glass (2001), *there is still room for systemic analysis of IT failures and there are very few accounts of errors of omission* in software projects.

- *Change the attitudes of clients in managerial and operational user roles*—viable research and practical activities in this direction could *use the work system method* (Alter, 2006) and other relevant methods to develop better understanding of organisational problems and to improve their communication with software developers.

3. **Translate agile and software issues into management and customer language:** We may suggest several possible directions here:
   - *Investigate in a systemic way the existing agile and plan-driven models* for software development and continue with the work started in Boehm (2006a) on creating new process models integrating not just SE and systems engineering ideas but other applicable systems concepts as well.
   - *Explore the applicability of "Sysperanto"* (see Alter, 2007) to foster a common language for all stakeholders in software development.
   - *Build methods and tools to facilitate the communication process between software developers, customers, and supporting multiple perspective* representations of problem situations as proposed by Linstone (1984).
4. **Emphasise value for every stakeholder:** Design science research and agile methods place high emphasis on this idea. There is a *need for more research on systemic identification of stakeholder values*. Further, there is a need for research on methods to model and help the effective analysis and *better systemic understanding of all aspects of software development*, related to the technical product attributes, the project organisational attributes, the developers attributes, and the client features in a particular project or system-of-projects.
5. **Pick good people, reward the results, and reorient the reward system to recognise both individual and team contribution:** These suggestions can be categorised as human resource management issues and hence are also suitable for *investigation through suitable systemic approaches and problem structuring methods, including multicriteria decision analysis, promoting evaluation, and decision making*.

One of the limitations of the scope of our proposals is that we have provided suggestions reflecting only on the above five ideas by Boehm and Turner (2005). A systemic investigation of all aspects of ISD could lead to a much broader set of considerations integrating SE, IS, and systems thinking. We believe, however, that the examples we have provided here can lead to easier adaptation and development of other relevant ideas serving a similar purpose. Another possible limitation is that we have produced our suggestions for future research on integrating SE, IS, and the systems approach by assuming that the current state of the art and practice in SE and IS are known and we have focused rather only on identifying examples of the use of a systems approach in IS or SE. As we have pointed out earlier, we have relied on the comprehensive analysis of the state-of-the-art of the IS discipline provided by Hirschheim and Klein (2003). We have also reflected on trends in SE (see Reifer, 2003; Boehm, 2006a, b; Boehm & Turner, 2004) and on the comparative analysis of research in the three computing disciplines by Glass et al. (2004). It would be interesting to conduct a further investigation of IS implementation as a whole that goes beyond the existing disciplinary boundaries and takes a systems approach as an organising viewpoint.

Most of our recommendations on integrating IS, SE, and systems thinking relate to issues

of organisational learning where contemporary systems methods have a significant history of achieving improvement. The challenge for IS and SE practitioners, researchers, and educators is not just to investigate the issues we discussed in this article but also to practice what was learned for improved ISD.

## ACKNOWLEDGMENT

The authors are very grateful to D. Bustard and I. Bider for their very insightful comments that helped improve the article and to the editors of IJITSA for their encouragement.

## REFERENCES

Abdel-Hamid, T.K., & Madnick, S.E. (1991). *Software project dynamics.* Englewood Cliffs, NJ: Prentice Hall.

Ackoff, R. (1999). *Ackoff's best: His classic writings on management*. New York: Wiley.

Ackoff, R. (2006). Why few organizations adopt systems thinking. *Systems Research and Behavioral Science, 23*(5), 705-708.

Alter, S. (2002). The work system method for understanding information systems and information systems research. *Communications of the AIS, 9*(6), 90-104.

Alter, S. (2004). Desperately seeking systems thinking in the information systems discipline. In *Proceedings of Twenty-Fifth International Conference on Information Systems* (pp. 757-769).

Alter, S. (2006). *The work system method: Connecting people, processes, and IT for business results*. Lakspur, CA: Work System Press.

Alter, S. (2007). Could the work system method embrace system concepts more fully? *Information Resources Management Journal, 20*(2), 33-43.

Argyris, C., & Schon, D.A. (1978). *Organizational learning: A theory of action perspective*. Addison-Wesley.

Aurum, A., & Wohlin, C. (Eds.). (2005). *Engineering and managing software requirements*. Heidelberg, Germany: Springer.

Avison, D. (2000). *Multiview: An exploration in information systems development* (2nd ed.). Alfred Waller Ltd.

Avison, D.E., & Fitzgerald, G. (2003). Where now for development methodologies? *Communications of ACM, 46*(1), 79-82.

Bajaj, A., Batra, D., Hevner, A., Parsons, J., & Siau, K. (2005). Systems analysis and design: Should we be researching what we teach? *Communications of the AIS, 15*, 478-493.

Baresi, L., Di Nitto, & Ghezzi, C. (2006). Toward open-world software: Issues and challenges. *IEEE Computer, 39*(10), 36-43.

Barton, J., Emery, M., Flood, R.L., Selsky, J., & Wolstenholm, E. (2004). A maturing of systems thinking? Evidence from three perspectives. *Systemic Practice and Action Research, 17*(1), 3-36.

Baskerville, R., & Wood-Harper, A.T. (1998). Diversity in information systems action research methods. *European Journal of Information Systems, 7*(2), 90-107.

Benbasat, I., & Zmud, R. (2003, June). The identity crisis within the IS discipline: Defining and communicating the discipline's core properties. *MIS Quarterly, 27*(2), 183-194.

Bennetts, P.D.C., Wood-Harper, T., & Mills, S. (2000). A holistic approach to the management of information systems development. *Systemic Practice and Action Research, 13*(2) 189-206.

Boehm, B. (2006a). Some future trends and implications for systems and software engineering processes. *Systems Engineering, 9*(1), 1-19.

Boehm, B. (2006b). A view of 20th and 21st century software engineering. In *Proceeding of the 28th international conference on Software Engineering.* Shanghai, China (pp. 12-29).

Boehm, B. (2006c). Educating students in value-based design and development (Keynote address). In *Proceedings of the 19th Conference on Software Engineering Education and Training (CSEET).*

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Boston: Addison-Wesley.

Boehm, B., & Turner, R. (2005). Management challenges for implementing agile processes in

traditional development organizations. *IEEE Software, 5*, 30-39.

Brooks, F.P. (1987). No silver bullet: Essence and accidents of software engineering. In *Proceedings of the IFIP 10th World Computing Conference* (pp. 1069-1076).

Bustard, D.W., & Keenan, F.M. (2005, April 3-8). Strategies for systems analysis: Groundwork for process tailoring. In *Proceedings of 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2005),* Greenbelt, Maryland (pp. 357-362).

Checkland, P. (1999). *Systems thinking, systems practice*. Chichester: Wiley.

Checkland, P., & Holwell, S. (1998). *Information, systems and information systems*. Chichester: Wiley.

Checkland, P., & Poulter, J. (2006). *Learning for action: A short definitive account of soft systems methodology and its use by practitioner, teachers and students.* Chichester: Wiley.

Chujo, H., & Kijima, K. (2006). Soft systems approach to project-based education and its practice in a Japanese university. *Systems Research and Behavioral Science, 23*(1), 89-106.

Cornford, T., & Smithson, S. (1996). *Project research in information systems.* MacMillan.

Crnkovic, I., Land, R., & Sjögren, A. (2003, March). Is software engineering training enough for software engineers? In *Proceedings of the 16th International Conference on Software Engineering Education and Training*, Madrid, Spain. IEEE.

Denning, P.J. (2005). Recentering computer science. *Communications of ACM, 48*(11), 15-19.

Denning, P.J., & Dunham, R. (2006). Innovation as language action. *Communications of ACM, 49*(5), 47-52.

Developer. (2007). Developer: An online magazine for software developers. Retrieved July 12, 2007, from *http://www.developerdotstar.com/mag/categories/systems_software_series.html*

Dietrich, Y., Floyd, C., & Klichewski, R. (2002). *Social thinking-software practice*. Boston: MIT Press.

Eom, S. (2000). The contribution of systems science to the development of the decision support systems subspecialties: An empirical investigation. *Systems Research and Behavioral Science, 17*, 117-134.

Finnegan, P., Galliers, R.D., & Powell, P. (2002). Planning electronic trading systems: Re-thinking IS practices via triple loop learning. In S. Wrycza (Ed.), *Proceedings of the 10th European Conference on Information Systems* (pp. 252-261). Retrieved July 12, 2007, from http://www.csrc.lse.ac.uk/asp/aspecis/20020114.pdf

Flood, R.L., & Romm, N.R.A. (1996). *Diversity management: Triple loop learning.* Chichester: Wiley.

Glass, R. (2001). *Computing failure.com*. Upper Saddle River, NJ: Prentice Hall.

Glass, R. (2005). Never the CS and IS Twain shall meet? *IEEE Software*, pp. 120-119.

Glass, R., Ramesh, V., & Vessey, I. (2004). An analysis of research in computing disciplines. *Communications of ACM, 47*(6), 89-94.

Gonzales, R. (2005, March-April). Developing the requirements discipline: Software vs. systems. *IEEE Software*, pp. 59-61.

Hevner, A.R., March, S.T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75-105.

Hirschheim, R., & Klein, H.K. (2003). Crisis in the IS field? A critical reflection on the state of the discipline. *Journal of the Association of Information Systems, 4*(5), 237-293.

Holwell, S. (2000). Soft systems methodology: Other voices. *Systemic Practice and Action Research, 13*(6), 773-797.

Hughes, J., & Wood-Harper, T. (1999). Systems development as a research act. *Journal of Information Technology, 14*, 83-94.

IbisSoft. (n.d.). Environment that promotes IS research. Retrieved July 12, 2007, from *http://www.ibissoft.se/english/index.htm?frameset=research_frame.htm&itemframe=/english/about_isenvironment.htm*

Iivari, J., Hirschheim, R., & Klein, H. (2004). Towards a distinctive body of knowledge for information systems experts: Coding ISD process knowledge in two IS journals. *Information Systems Journal, 14*, 313-342.

Iivari, J., & Huisman, M. (2007). The relationship between organizational culture and the deployment of systems development methodologies. *MIS Quarterly, 31*(1), 35-58.

Jackson, M.C. (1992). An integrated programme for critical thinking in information systems research. *Journal of Information Systems, 2*, 83-94.

Jackson, M.C. (2003). *Systems thinking: Creative holism for managers.* Chichester: Wiley.

Jackson, M C. (2006). Creative holism: A critical systems approach to complex problem situations. *Systems Research and Behavioral Science, 23*(5), 647-657.

Kroes, P., Franssen, M., van de Poel, I., & Ottens, M. (2006). Treating socio-technical systems as engineering systems: Some conceptual problems. *Systems Research and Behavioral Science, 23*(6), 803-814.

Kruchten, P. (2005, March-April). Casting software design in the function-behavior-structure framework. *IEEE Software*, pp. 52-58.

Kuhn, T.S. (1970). *The structure of scientific revolutions* (2nd ed.). Chicago: University of Chicago Press.

Lane, J.A., & Boehm, B. (2007). System-of-systems cost estimation: Analysis of lead system integrator engineering activities. *Information Resources Management Journal, 20*(2), 23-32.

Laware, J., Davis, B., & Peruisch, K. (2006). Systems thinking: A paradigm for professional development. *The International Journal of Modern Engineering, 6*(2).

Lee, A. (2000, May). Systems thinking, design science, and paradigms: Heeding three lessons from the past to resolve three dilemmas in the present to direct a trajectory for future research in the information systems field *(*Keynote address). In *Proceedings of the 11th International Conference on Information Management*, Taiwan. Retrieved July 12, 2007, from http://www.people.vcu.edu/aslee/ICIM-keynote-2000

Lee, A. (2004). Thinking about social theory and philosophy for information systems. In J. Mingers & L. Willcocks (Eds.), *Social theory and philosophy for information systems* (pp. 1-26). Chichester: Wiley.

Lee, B., & Miller, J. (2004). Multi-project software engineering analysis using systems thinking. *Software Process: Improvement and Practice, 9*(3).

Lewis, P. (1995). New challenges and directions for data analysis and modeling. In F. Stowell (Ed.), *Information systems provision: The contribution of soft systems methodology* (pp. 186-205). London: McGraw-Hill.

Linstone, H.A. (1984). *Multiple perspectives for decision making: Bridging the gap between analysis and action.* New York: North Holland.

Madachy, R.J. (2006). Integrated modeling of business value and software processes. *Lecture Notes in Computer Science, 3840*, 389-402.

Madachy, R.J. (2007). *Software process dynamics.* Wiley/IEEE Press.

Madachy, R.J., Boehm, B., & Lane, J.A. (2007). Software lifecycle increment modeling for new hybrid processes. In *Software process improvement and practice.* Wiley. Retrieved July 12, 2007, from http://dx.doi.org/10.1002/spip.332

Mason, R., & Mitroff, I. (1981). *Challenging strategic planning assumptions.* New York: Wiley.

Mathieu, R.G. (2002). Top-down approach to computing. *IEEE Computer, 35*(1), 138-139.

McLeod, R. (1995). Systems theory and information resources management: Integrating key concepts. *Information Resources Management Journal, 8*(2), 5-14.

Mingers, J. (1995). Using soft systems methodology in the design of information systems. In F. Stowell (Ed.), *Information systems provision: The contribution of soft systems methodology* (pp. 18-50). London: McGraw-Hill.

Mingers, J. (2001). Multimethodology: Mixing and matching methods. In J. Rosenhead & J. Mingers (Eds.), *Rational analysis for a problematic world revisited*. Chichester: Wiley.

Mitroff, I., & Linstone, H. (1993). *The unbounded mind.* New York/Oxford: Oxford University Press.

Mora, M., Gelman, O., Forgionne, G., Petkov, D., & Cano, J. (2007). Integrating the fragmented pieces of IS research paradigms and frameworks: A systems approach. *Information Resource Management Journal, 20*(2), 1-22.

Ngwenyama, O.K., & Lee, A.S. (1997). Communication richness in electronic mail: Critical social theory and the contextuality of meaning. *MIS Quarterly, 21*(2), 145-167.

Petkov, D., Petkova, O., Andrew, T., & Nepal, T. (in print). Mixing multiple criteria decision making with soft systems thinking techniques for decision support in complex situations. *Decision Support Systems*.

Petkova, O., & Roode, J.D. (1999). An application of a framework for evaluation of factors affecting software development productivity in the context of a particular organizational environment. *South African Computing Journal, 24*, 26-32.

Pfleeger, S.L. (2001). *Software engineering theory and practice.* Upper Saddle River, NJ: Prentice Hall.

Pressman, R. (2001). *Software engineering: A practitioner's approach* (5th ed.). New York: McGraw-Hill.

QAA. (2007). *Subject benchmark statements: Computing*. Quality Assurance Agency. Retrieved July 12, 2007, from http://www.qaa.ac.uk/academicinfrastructure/benchmark/statements/

Reifer, D. (2003, November-December). Is the software engineering state of the practice getting closer to the of the art? *IEEE Software, 20*(6), 78-83.

Richmond, B. (1994, July). System dynamics/systems thinking: Let's just get on with it. In *Proceedings of the 1994 International System Dynamics Conference*, Sterling, Scotland. Retrieved July 12, 2007, from http://www.hps-inc.com/st/paper.html

Saaty, T. (1990). *Multicriteria decision making: The analytic hierarchy process* (2nd ed.). Pittsburgh: RWS Publications.

Sage, A.P. (2005). Systems of systems: Architecture based systems design and integration (Keynote address). In *Proceedings of the International Conference on Systems, Man and Cybernetics*, Hawaii.

Sewchurran, K., & Petkov, D. (2007). A systemic framework for business process modeling combining soft systems methodology and UML. *Information Resources Management Journal, 20*(3), 46-62.

Sommerville, I. (2007). *Software engineering* (8th ed.). Harlow: Pearson.

Stowell, F. (Ed.). (1995). *Information systems provision: The contribution of soft systems methodology.* London: McGraw-Hill.

Stowell, F., & West, D. (1996). Systems thinking, information systems practice. In *Proceedings of the 40th Conference of the International Society for Systems Science*s, Budapest, Hungary.

SWEBOK. (2004). Software engineering body of knowledge defined by the IEEE CS and ACM. Retrieved July 12, 2007, from http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.pdf

Sweeney, A., & Bustard, D.W. (1997). Software process improvement: Making it happen in practice. *Software Quality Journal, 6*, 265-273.

Ulrich, W. (1998). *Systems thinking as if people mattered: Critical systems thinking for citizens and managers* (Working paper no. 23). Lincoln School of Management.

Valerdi, R., & Madachy, R. (2007). Impact and contributions of MBASE on software engineering graduate courses. *Journal of Systems and Software, 80*(8), 1185-1190.

Van Vilet. (2000). *Software engineering: Principles and practices.* Wiley.

Vo, H.V., Paradice, D., & Courtney, J. (2001). Problem formulation in inquiring organizations: A multiple perspectives approach. In *Proceedings of the 7th Americas Conference on Information Systems*, Boston, Massachusetts.

Vo, H.V., Chae, B., & Olson, D.L. (2006). Integrating systems thinking into IS education. *Systems Research and Behavioral Science, 23*(1), 107-122.

Weinberg, G. (1992). *Quality software management* (Vol. 1: Systems Thinking). New York: Dorset House Publishing.

Weinberg, G. (2007). A site for books, articles and courses. Retrieved July 12, 2007, from http://www.geraldmweinberg.com/

Wendorff, P. (2002). Systems thinking in extreme programming. In *Proceedings of the 10th European Conference on Information Systems* (pp. 203-207). Retrieved July 12, 2007, from http://www.csrc.lse.ac.uk/asp/aspecis/20020124.pdf

White, L., & Tacket, A. (1996). The end of theory? *Omega, 24*(1), 47-56.

Wilson, B. (1990). *Systems: Concepts, methodologies and applications* (2nd ed.). Chichester: Wiley.

Xu, L.D. (2000). The contribution of systems science to information systems research. *Systems Research and Behavioral Science, 17*(2), 105-116.

Yourdon, E. (2007). A site for books, articles and blogs. Retrieved July 12, 2007, from http://www.yourdon.com

Zhu, Z. (2006). Complementarism versus pluralism: Are they different and does it matter? *Systems Research and Behavioral Science, 23*(6), 757-770.

*Doncho Petkov is a professor and coordinator in IS at Eastern Connecticut State University, USA. He has taught at university level since 1987 and prior to that he worked in the IT industry for nine years. He is a senior area editor for* IJITSA*, co-editor of* IJCSS *and a member of the editorial boards of* Systems Research and Behavioral Science and Scientific Inquiry*. His publications have appeared in the* Journal of Systems and Software, Decision Support Systems, Telecommunications Policy, JITTA, International Journal on Technology Management, Kybernetes, JITCAR *and elsewhere.*

*Denis Edgar-Nevill is head of the Department of Computing at Canterbury Christ Church University in the United Kingdom. He has held academic posts in computing at four UK universities over his 28 year career. He has led three European Union funded projects involving university and commercial partners across Europe and organized and hosted 16 international conferences in the areas of software quality, information systems, and cybercrime forensics. He has also been a member of international advisory panels for 4 other major international conferences. He is a Fellow of the British Computer Society and a member of the BCS Elite Group.*

*Raymond Madachy is a research assistant professor in the Epstein Department of Industrial and Systems Engineering at the University of Southern California and Principal of the USC Center for Systems and Software Engineering. He is currently serving as interim director of the Systems Architecting and Engineering Program and has 25 years of management and technical experience in industry. His research interests are in modeling and simulation of processes for architecting and engineering of complex software-intensive systems. He is a co-author of the book* Software Cost Estimation with COCOMO II *and his book* Software Process Dynamics *will be available in late 2007.*

*Rory O'Connor is a senior lecturer in software engineering at Dublin City University and a senior researcher with Lero, The Irish Software Engineering Research Centre. He has previously held research positions at both the National Centre for Software Engineering and the Centre for Teaching Computing, and has also worked as a software engineer and consultant for several European technology organizations. His research interests are centered on the processes whereby software intensive systems are designed, implemented and managed.*