Key issues in Information Systems and Software Engineering: 
views from a joint network of practitioners and academics

R. HARRISON  
School of Computer Science, Cybernetics & Electronic Engineering  
Department of Computer Science, University of Reading, UK

D. E. AVISON  
Department SID, ESSEC Business School, France

K. H. BENNETT  
Department of Computer Science,  
University of Durham, UK

R. BISHOP  
GID Ltd, UK

M. RAMAGE  
Centre for Complexity and Change, The Open University, UK

D. TARGETT  
Imperial College Management School, UK

1. Introduction

This paper reflects on the opinions of the SEISN (Software Engineering and Information Systems Network for improved business processes) on a number of key issues. The main stimulus for the formation of the network came from the lack of communication between the software engineering (SE) and information systems (IS) communities. Software engineering can be defined as the establishment and use of sound engineering principles in order to obtain efficient software in terms of cost and reliability. The study of information systems on the other hand concerns the application of information technology to business problems including the human and organisational aspects of software use. One of the aims of the network was to encourage multidisciplinary and interdisciplinary work for the benefit of both the SE and IS communities. The SEISN brought together leading researchers and practitioners in computer science, information systems and management to form a community which could explore the links between the disciplines in academia and outside, debating conflicts and misunderstandings, proposing best practice and disseminating this to academia and industry. The network focused on the exchange of ideas between the software engineering and information systems communities, enabling these communities to clarify their beliefs and present experiences, findings and views to both communities at large. In this paper, we try to investigate where there is common ground between the IS and SE communities and practitioners and where differences remain.

The SEISN organised a series of nine workshops throughout the past three years, stimulating some very constructive interaction. Many topics were debated, but this paper reflects the opinions of the SEISN on four topics that were seen to be key issues for SE and IS for this decade: rapid application development, managing information and knowledge, co-evolution and the future of business and IT.

We begin by looking at issues related to the attempt to develop applications more rapidly. Although ‘Rapid Applications Development’ (RAD) is associated with a movement and IS development approach of the early 1990s [14] the aim of developing applications rapidly (but also cheaply and of high quality) is enduring. Indeed, the wish to have an Internet presence and develop web applications quickly has furthered the requirement for quick applications development. Changes in the industry and business
environment (increased competition, globalisation, enterprise resource planning systems, e-business, ‘speed to market’ as well as a general desire to stay with the ‘leading edge’) are all business drivers for rapid application development. Frequently technologists hear ‘requirements’ expressed as ‘I am not sure what I need, but I want it now’. This has also meant that the lifetime of software is not expected to be as long as in the past, indeed ‘legacy systems’ might simply be replaced so that the term may well go out of fashion.

In the second section we explore the reasons why knowledge and its management have become a focus of attention for managers, researchers and pundits. We also discuss what we mean by knowledge, and methods of knowledge management. Knowledge is replacing information as one of the most important areas that potentially give organisations competitive advantage. Businesses compete daily on the basis of knowledge, and knowledge lies behind an organisation’s ability to maintain its success, innovate, deliver customer value and operate effectively. Knowledge is behind the idea that organisations should be learning environments where people and the organisation as a whole are continually developing and changing. At the same time IS have become all pervasive. In particular the Internet has an enormous capacity to create, store, process and communicate information, and this is a short step from knowledge. Other factors are also relevant. For example, downsizing has heightened the need to retain knowledge which otherwise would be lost as people leave; the trend towards focussing on customers has increased the demand for information and knowledge.

The third section of this paper explores co-evolution. Business and IT evolve but often not in ways that complement each other. More synergy and co-operation is needed to ensure co-evolution. There is a need to change business processes because of the rapid rate of change of IT but large changes in technology bring complications for business. Virtual working is a good example of a new business process brought about by changes in IT. Unfortunately engineers often do not understand co-evolution: instead we have managed chaos. This may be because it is often difficult to get customers to articulate what they need and how they work. Indeed, emergent organisations may not even have a clearly defined business process. Many businesses do not act strategically and in fact managers often complain that IT decisions are imposed on them.

The final section of this paper looks to the future, and particularly at the dynamics of business and IT. There is a well-established evolutionary route in many domains (beyond IT) in which small companies which are successful evolve into large companies. New technical ideas are often injected into this process through the highly innovatory starter company. Alternatively, promising small businesses may be purchased by larger organisations, so that the latter can benefit quickly in the marketplace from the innovation. In due course, the markets may change and the large organisation may fragment, change out of all recognition, or even fail (this is a risk of failing to innovate). Over 40% of the top Fortune 500 US corporates in business in 1979 no longer exist in corporate form. In the EU, 30% of firms with less than ten employees generate 70% of EU turnover. We look at the challenges that this brings business and IT.

2. Developing Applications Rapidly

2.1 EXtreme Programming

In the age of internet time, pressures are on business to deliver applications rapidly. Fortunately there are technology drivers, new agile software development methodologies such as eXtreme Programming (XP), that support faster development of software, particularly for small and medium-sized applications and organisations. Jeffries defines XP as ‘a discipline of software development with values of simplicity, communication, feedback and courage.’ [13]. The four project values of XP are Simplicity, Communication, Testing and Aggressiveness. Beck also cites these values [5].

A number of organisations are utilizing XP in their software development to reap its potential efficiency gains. XP is a lightweight methodology that places emphasis on
teamwork. The team consists of customer, management and developers. XP convenes these people and assists them in succeeding together.

The customer must define their requirements in user stories, which define what the system needs to do for its users. A crucial component of defining a user story is the definition of test scenarios: testing is always up-front in XP. Design problems are solved via architectural spikes: simple prototype systems that can be used to explore potential solutions. The spike only addresses the problem under examination and ignores all other concerns. A spike is not high-quality software and is usually discarded. The purpose behind the use of a spike is to reduce the risk of a technical problem or increase the reliability of a user story.

Pair programming reduces the potential risk when a technical difficulty threatens to hold up the system's development. A release plan is used to plan for each iteration, lasting one to three weeks. In iteration planning acceptance tests are created from user stories and scheduled. The customer specifies scenarios to test when a user story has been correctly implemented. A story can have one or many acceptance tests, as necessary.

2.2 Alternatives

Of course there are alternative approaches to developing applications quickly, as well as other agile approaches. XML represents a dynamic database evolutionary approach; quick end-user computing is feasible if the application falls easily within the parameters of, for example, a spreadsheet, Access database solution or web development tools. Also, linear workflow systems and other application packages can be bought off the shelf and remain a good solution if they do not require much modification. In addition the Capability Maturity Model can be a helpful support as it encourages reflection. A lot, therefore, has been achieved but much still remains to be done.

2.3 No Silver Bullets

The relative lack of success of CASE and other tools, reusable component development and other silver bullets has left a legacy of disillusionment. It is therefore important that XP is seen as ‘potentially helpful’ and not as ‘the answer’. Relational database systems, data modelling, and methodologies such as SSADM [3] and other traditions may slow rather than speed up the process. Further, large organisations simply cannot change fast. But, the comparative slowness of requirements analysis might be due to the complexity of the application and not necessarily due to inappropriate techniques. Many organisations do not have a great deal of slack that can be used for thinking and analysis [11].

2.4 Quality

In addition to these inhibitors there are genuine concerns over quality that must not be forgotten. Software and IS development must not make speed of development an overriding concern at the expense of quality, good project management procedures and what trust has been achieved over the last few years between change agents and their clients. But clients need also to realise that quality and speed of development may conflict. However, notwithstanding these provisos, there are business drivers, software and IS facilitators that may lead to a balance of both quick and effective applications development, or at least movements in this direction.

3. Managing information and knowledge
3.1 What is knowledge?

The distinction between three types of intellectual asset (data, information and knowledge) has been recognised by many writers\(^1\).

- **Data** are merely records of facts, transactions or forecasts with no attempt at interpretation. Data are raw numbers and are easily collected.
- **Information** is data structured to be meaningful, for example a supermarket may process its data to provide information about the sales levels of different items at different stores.
- **Knowledge** is information processed by individuals, giving an interpretation of a situation; people add value to information creating knowledge [8]). Knowledge is the “larger, longer-living structures” of information [7].

A more poetic version of this distinction comes from the process of wine-making. Information could be thought of the grapes, yeast and sugar, while knowledge could be thought of as the final product. Wine obviously needs some kind of container to hold it and give it shape, and in this respect resembles knowledge more closely than a bottle or glass containing it.

Formal definitions also recognise the difference between explicit and tacit knowledge. Explicit knowledge can be represented in the form of formulas, diagrams and reports and is therefore relatively easy to capture and communicate. On the other hand tacit knowledge is embedded in peoples' minds and impossible to represent in such ways. For example, a management consultant’s ability to approach a complex assignment is tacit knowledge; an owner’s manual accompanying the purchase of an electronic product is explicit knowledge. This distinction affects the transferability of knowledge and how it is done. Alavi and Leidner [1] point out the following paradox: tacit knowledge is the necessary background for interpreting explicit knowledge. Only individuals with overlaps in their underlying knowledge bases (a shared knowledge space) can truly communicate and exchange knowledge. People lacking this knowledge, those who would most benefit from the exchange, cannot join in until they can enter this shared space. Social rather than technological methods are needed to bring this about. Consequently knowledge management should not be reduced to the problem of developing an IS for exchanging information. Many organisations, however, seem to adopt such an approach.

3.2 Knowledge Management

There are two main methods of managing knowledge: technical and social. Technical methods include:

- Artificial intelligence approaches which aim to mimic human behaviour.
- Decision support systems such as the use of credit scoring.
- Groupware (particularly tools such as Lotus Notes.)
- Data mining tools such as those used extensively for customer relationship management.

Social methods include:

- Creating spaces for informal knowledge sharing (such as areas for photocopiers and coffee machines.)
- Events which bring together people from across an organisation.

---

\(^1\) For example, [7] [8], and [16].
• Apprenticeship and mentoring processes (cf. [18])
• Storytelling.
• Censorship (whether imposed by the self or externally.)
• Unlearning.
• Crisis management.

These two distinct aspects of knowledge management seem to be handled quite differently in many organisations. Clearly there needs to be considerable interdependence between them if they are to be useful. The distinction, shown in Figure 1, could be considered as being between ways of holding/storing knowledge (technical methods) and ways of sharing knowledge (social methods). Of course this is not an orthogonal separation; Lotus Notes for example plays a role in both aspects.

The simplistic split of Figure 1 implies that between the two basic activities of holding/storing knowledge and sharing knowledge are a set of other processes: creating it, embodying it in some physical form, and manipulating it. Figure 2 shows this cycle of different activities. However, instead of being simply cyclic, the process of going around this cycle leads to a different form of knowledge (in principle better or deeper, but definitely different) arising at each iteration. This is the point also made by Nonaka and Takeuchi [7] in their consideration of a “knowledge spiral”. Given the spiral nature of the process, the slowness with which real knowledge arises (as opposed to data or information) and our earlier analogy of the maturing of wine, we are reminded of a snail (see figure 2).

We recognise that using diagrams such as Figures 1 and 2 to summarise the knowledge process may give the impression that knowledge is something that can be touched and pinned down. We know that this is not the case, as the distinction between tacit and explicit knowledge makes clear.
3.3 The Dynamics of Knowledge and Knowledge Management

Knowledge changes over time. It is clear that if something can still be said to be knowledge, it does not simply decay, but rather changes into a different form. The mechanism by which it changes (the longer-term version of our knowledge snail) is debatable. Is it Darwinian, an evolutionary process based on the survival of the "fittest ideas" and "fittest worldviews" (cf. Dawkins’ [10] concept of the “meme”, a unit of knowledge behaving like a gene)? Or is it more Foucauldian, in that changes in knowledge stem from who has the power to pass on ideas and shape others’ understandings (cf. the African proverb that “until lions have historians, history will always favour the hunter”)? In either case, one clear way in which knowledge is shared and renewed is the traditional concept, currently much in vogue within organisational studies, of storytelling.

4. Co-evolution of business processes and technology

4.1 Models of co-evolution

There are two sorts of relationships that business has with IT: tight coupling, where the core business is intimately related to the IT system (e.g. Amazon), and loose coupling, where the IT system is almost completely unrelated to the core business (e.g. a payroll system). Co-evolution is essential for tightly coupled systems, but less so for loosely coupled ones.

However, there is no single model of co-evolution. We see co-evolution via osmosis as one of the more successful models for small and medium size enterprises. Local co-evolution is clearly easier than global co-evolution, but localised changes may lead to legacy systems and security loopholes. Global solutions can also be beneficial. An agile business is more likely to lead to customer satisfaction and is easiest to achieve with small businesses that are pro-active. Uncertainty in business is easiest to manage locally.

Three different situations for co-evolution are shown in Figures 3-5 in which weight is depicted as analogous to evolutionary change. Figure 3 shows a stable situation with IT in control. As business evolves the situation will become unstable. This situation arises if there is a delay as far as feedback is concerned. Figure 4 shows a stable situation with shared control. This is resource hungry, as it requires constant feedback. If either business or IT evolve alone then the situation becomes unstable. For stability, changes in IT must be reflected by changes in business and vice-versa. Figure 5 shows a stable situation with business in control. If IT evolves then the situation is bound to change.

Figure 3
4.2 Facilitators and inhibitors of co-evolution

Feedback is absolutely crucial for successful co-evolution. Hysteresis (the delay of feedback from business to IT and vice-versa) is an inhibitor of co-evolution. An absence of feedback is extremely damaging to co-evolution, and even delayed feedback produces unexpected results. Communication is one of the keys to co-evolution, as are people. Unfortunately the business and IT communities use different languages. We suggest a Chief Information Officer could play a key role in facilitating co-evolution. Co-location facilitates co-evolution, as dissemination of information is easier for co-located teams. Both bottom up and top down decision making are necessary for co-evolution (top-down particularly for strategic decisions).

Understanding customers and their needs is very important. Business people should educate or liaise with technologists. Conversely, business people need to understand what IT can provide. Without such understanding, requirements may be unrealistic.

The Web is an example of the way that business processes have changed because the supporting IT infrastructure evolved. The prevalence of digital libraries rather than real-world libraries is another example of evolving processes. The Boeing Corporation is a business in which the core skill is project management. Consequently managers need control of the project management software, but the business and IT systems do not necessarily need to co-evolve.

4.3 Research Directions

Co-evolution is poorly understood by both managers and IT experts alike. Research into the facilitators and inhibitors of co-evolution is needed. An investigation into the application domains for which co-evolution is essential is also important. We should also consider whether co-evolution is desirable in all situations. Communication methods and knowledge management, particularly for distributed working, also need further investigation.

5. Prospects for the future of business and IT

5.1 Business Dynamics

In IT, it is not clear what business dynamics apply. IT is clearly a big business; the estimate of world annual IT revenue for 2002 is around $250 Billion [12]. The analysis of the business dynamic, through size, efficiency, change and domains, and its link to technology change, through complexity, evolution and timescales is very broad. Of
particular interest however is the technical innovation process and its links to academic research. In very broad terms, research in IT can be seen to have two complementary and mutually reinforcing strands:

i) Product-based innovation, leading directly to better (i.e. faster, more usable, cheaper, more reliable) products; an example could be a faster speech recognition algorithm.

ii) Process-based research, where better processes lead to better ways of making software (for example, lightweight software engineering process models)

The IT scene in key market domains is dominated by major US companies. These organisations (through shrink-wrapped products) really determine what users actually receive and use. For many, this is exactly what is wanted. However, it is not clear how the next brilliant idea is taken up and used (to help users) within this environment. Research transitioning from academic work to industry can be stifled, and exciting new ideas crushed.

5.2 The Challenge

A recent study [6] has shown that users may have profound and fundamental problems with the software that they currently employ. The study highlighted five basic concerns:

•Necessary and sufficient: the software should do what the user requires, and not be overburdened with extra over-engineered and unwanted facilities that are not needed. Many users do not touch many of the facilities available in modern packages, and may be bewildered by the complexity.
•Personalisation: the software should be capable of simple alteration to provide a working environment to meet the user’s profile and level of skill and needs, instead of the generic approach common in most products.
•Adaptable/self-adaptation. The software should not need to be maintained explicitly to keep up with changing requirements. For example, the software itself could commission extra required functionality and decommission unwanted or redundant features.
•Distribution and granularity: software at present is delivered in very large discrete units; it would be better to build up functionality from small grain components that meet exact needs. Future software should be structured in small simple units that co-operate through rich communication structures and information gathering. This will provide a high degree of resilience against failure in part of the software network and allow software to re-negotiate use of alternatives. It also stimulates a different business model for delivering the software.
•Transparency: software should be seen as a single abstract object even when distributed across geographical locations and different platforms (see also [2]). This is an essential property if the software is able to reconfigure itself and substitute one component or network of components for another. This should include issues such as security, code and data mobility.

5.3 Paradigm Shifts

In the past paradigm shifts have led to severe restructuring in the computing industry; the most obvious recent example is the change from central mainframe computing to PC type workstations. Within the next 5 - 15 years we envisage that the following technology changes could have a major impact:

i) The open source movement.

ii) Bio-computing.
iii) Quantum computing.
iv) Ubiquitous computing and universal networking.

Another more radical possibility is a move “back to nature” which rejects technology in everyday life.

We feel that two “paradigm shifts” will have the biggest effect in the short to medium term. The first is a technical issue: the open source movement. That such software is free of charge is but one issue. Open source software is gaining an ever more successful track record for reliability and resilience, and may have an impact on security and cryptography, where full disclosure of algorithms and vulnerabilities could play an important role, as reported on a website [19]. There are very few parallels in other technological domains. The second (which is linked) is a move away from a business model in which software is sold and owned, as a shrink wrapped package, or bespoke system, to a model in which software is used as a service. In the second case, charging might be on a per-use basis, so income by vendors moves from a capital to a revenue basis.

This could lead to a much more open market, without the domination of a few big US suppliers. If users could assemble just the software they need for their purposes (instead of what they are given), small businesses could find the means to enter the marketplace by selling carefully targeted niche services in particular domains.

There is some evidence that this approach is being taken seriously, but at present it is envisaged as a technical solution by major vendors (e.g. for constant evolution, or for revenue generation [18]). The hard problems come with multidisciplinary issues, such as payment mechanisms, legal and contractual matters, trust and confidence, market structures, service composition and so on. There is some evidence that users will resist a pay-per-play approach with its open-ended costs if there is an alternative based on predictable budgets. Open-source software becomes even more attractive, even if skilled staff have to be recruited to maintain it, as the costs are clearer. These terms and conditions pose many interesting challenges that will require collaboration between researchers in software engineering and information systems.

6. Conclusions

After summarising the findings for the four topics, the common threads and differences between IS, SE and practitioners will be highlighted. These threads indicate possible topics for future research.

Whilst rapid application development is strategically important, we need research to ensure that moving faster does not mean poorer quality applications. We need to study the human processes relating to faster development so that quality does not suffer. Further, faster software development is only part of the process, and thorough requirements definition, systems analysis and design may not be similarly speeded up in the overall process of application development. Some approaches, such as Multiview [4] and QuickETHICS [15] have attempted to address this issue along with RAD. Potential drivers here might be quick consensus development through better group decision support tools [8] and better reward systems. Having said this, however, there is a need for research that attempts to demonstrate convincingly that those companies that have moved quickly in developing their software applications have gained competitive advantage in the long term. There is evidence that early movers may gain in the short term, but this advantage may be short lived especially if fast development is not linked to better quality.

Turning to the need for future research on the socio-technical management of knowledge, open questions include:

- Linking knowledge management with organisational decision-making: how is it possible to represent knowledge in a way that enables people to act? For example,
does the term ‘The Knowledge’ used by taxi-drivers actually refer to knowledge or something else?

- What does it really mean to say that organisations have knowledge, or that they learn? Two perspectives of knowledge management which are worth developing are the work of Freud (viewing new learning not as unlearning but the accretion of new layers over time) and of Foucault (in relation to the changing of knowledge through power).
- Story-telling as a means of developing and transferring knowledge. As applied to legacy IT systems, this could be seen as a way of bringing out tacit knowledge; and as currently used in the Capability Maturity Model of software process improvement, as a way of assessing the level of an organisation’s maturity.

As far as co-evolution is concerned, there is no silver bullet simply because there are too many variables in real-world systems. There must be some slack in systems in order to manage co-evolution [11], as time to reflect is essential. Management of knowledge is also essential. Business processes and IT should be agile. Business should be supported, not undermined, by IT systems.

It is clear that IT and business face a number of challenges in the short to medium term. Two possible solutions have been presented: the open source movement and software as a service. We believe that both issues will be at the forefront of developments in IT and business.

An analysis of issues where there is common ground between IS, SE and practitioners reveals the following commonalities:

- Rapid application development is important for strategic advantage.
- Knowledge has a strategic role in business.
- Social methods of knowledge management are at least as important as knowledge management IS.
- The nature of the knowledge process (the knowledge snail of figure 2).
- Communication and timely feedback are essential.
- People form the very necessary interface between IT and business.
- Knowledge management is essential.

A similar analysis reveals the following differences:

- The SE community emphasises speed of delivery but the IS community considers that time spent on understanding the problem and requirements can not be compromised as this leads to speedy delivery of unsatisfactory systems rather than timely delivery of satisfactory systems.
- Definitions of terms: for example, what does constitute knowledge in the real world? Much terminology is dependent on context and environmental factors.
- The theoretical basis of knowledge management; is this Darwinian, Foucauldian or Freudian?

The SEISN succeeded in bringing together researchers and practitioners with a wide range of research interests, and the resulting discussions and collaborations have been varied and animated. We all agree on the importance of bringing together the software engineering and information systems communities and raising awareness of our work through publication of journal and conference papers, as well as through workshops such as those of the SEISN. However, it is clear that much remains to be done, particularly in the area of truly interdisciplinary research in which people from different disciplines work together to achieve a common goal.
Acknowledgements

The authors wish to acknowledge the following contributors to this paper: Paul Layzell (UMIST), Paul Dunning Lewis (Lancaster), Dani Rodriguez Garcia (University of Reading) and Mike Holcombe (University of Sheffield), Carole Brooke (University of Lincoln), Tim Millea (University of Reading), Ian Seward (Oxford Software Engineering Ltd) and David Targett (Imperial College), Helen Edwards (University of Sunderland), Stephen Cook (University of Reading) and Shoaib Qureshi (Philips Research Laboratories UK) Hussein Zedan (De Montfort University) and Frank Stowell (De Montfort University). The authors also wish to acknowledge the support of the UK EPSRC which has funded SEISN (GR/M02552) for this work. We would also like to thank all of those who participated in the SEISN events without whom this work would not have been possible. Our thanks go to all the SEISN committee members who devoted their valuable time and effort to make the network possible, and particularly to Professor Manny Lehman of Imperial College and his group for providing logistical support for our workshops. We would also like to thank all of the members who helped to make the network so successful.

References

2) ANSA – An engineer’s introduction to the architecture APM Ltd, Cambridge, 1989
19) http://research.microsoft.com/research/sn/Millennium/mgoals.html
20) http://www.counterpane.com/crypto-gram-0002.html#PublicizingVulnerabilities