How Visual Basic Entered the Curriculum at an Australian University: An Account Informed by Innovation Translation

Arthur Tatnall
Victoria University, Australia
Arthur.Tatnall@vu.edu.au

Bill Davey
RMIT University, Australia
Bill.Davey@rmit.edu.au

Abstract

In this paper the authors relate an example of an approach to conceptualizing curriculum innovation based on Innovation Translation, informed by Actor-Network Theory (ANT). This approach has an advantage over other innovation models in allowing the researcher to concentrate on just those aspects of the innovation that led to its adoption in a particular form, rather than relying on the explanatory power of its supposedly innate characteristics. The paper briefly outlines the theory of innovation translation, and actor-network theory, and describes an instance of how this theory can be applied to describing an information systems curriculum innovation. This example shows the advantages of innovation translation over other ways of viewing curriculum change.

Keywords: Information systems, curriculum, innovation, actor-network theory

Innovation and Change in Information Systems Curricula

That information systems (IS) curricula are under constant pressure to change is well known to all those involved in their delivery. The curriculum literature has a lot to do with innovation but makes little mention of the mechanisms by which innovation might occur. The adoption of Visual Basic into the IS curriculum of the Australian University described in this paper, was not just a minor curriculum change in which one programming language replaced another similar language, but represented an entirely new and different approach to programming. The dictionary defines an innovation as occurring when something new or different is introduced, and the adoption of Visual Basic at this university can thus certainly be considered as an innovation. This paper will examine the detail of how this innovation occurred.

Many of the reported studies on curriculum innovation are based on research, development and dissemination models (Havelock, 1971). Relying on logical and rational decisions, change models of this type depend on the use of convincing arguments based on programs of research. They posit a rational and orderly transition from research to development to diffusion to adoption (Kaplan, 1991), and it is unlikely that this approach has much relevance to university curricula (Tatnall, 2000). Problem solving models represent another approach to conceptualizing curriculum change in which this is seen as due to perceived educational need. Also a quite rational approach (Nordvall, 1982) they involve searching for alternative solutions, often by looking at what colleagues with similar interests are doing but sometimes by other forms of research, in an attempt to find a solution to the educational problem. Change is considered to occur in stages where needs are first identified and articulated as problems before solutions are sought, selected and applied.

An alternative view of innovation is proposed in actor-network theory (ANT) the cornerstone of which is translation (Law, 1992), which can be defined as: "... the means by which one entity gives a role to others." (Singleton & Michael, 1993 :229). This paper briefly describes the theory of innovation translation and examines an extended example of its use in explaining information systems curriculum change.

Innovation Translation and Actor-Network Theory

A common approach to researching innovation in disciplines such as Information Systems is to focus on the technical aspects of an innovation, and to treat 'the social' as the context in which its development and adoption take place (Tatnall & Gilding, 1999). Approaches of this type assume that outcomes of technological change are attributable to the 'technological' rather than the 'social' (Grint & Woolgar, 1997). At the other extreme social determinism holds that relatively stable social categories can be used to explain technical change (Law & Callon, 1988) and concentrates on the investigation of social interactions, relegating the technology to context; to something that can be bundled up and forgotten. This bundling means that fixed and unproblematic properties or 'essences' can then be assigned to the technology and used in any attempted explanation of change.

An important paradigm in innovation research is that of innovation diffusion: most studies of innovation in information systems and education making use of this approach. Innovation diffusion (Rogers, 1995) is based on the notion that technological innovations embody 'information': some capacity or 'essence' that is largely responsible for determining their rate of adoption. A significant problem with an essentialist paradigm like this arises when a researcher tries to reconcile the views of all parties involved in the innovation. The difficulty is that people often see *different* 'essential attributes' in any specific technological or human entity, making it difficult to identify and settle on the ones that allegedly were responsible for the diffusion.

Rather than relying on some 'inner technological logic', Brey (1997) proposes that technological change is best understood by reference to technological controversies, disagreements and difficulties with which the actors involved in the change are concerned. In a small step from this, Actor-Network Theory (ANT) considers both social and technical determinism to be flawed and proposes instead a socio-technical account (Callon, 1999; Latour, 1986; Law & Callon, 1988) in which neither social nor technical positions are privileged. It offers the notion of heterogeneity to describe socio-technical projects as this then avoids questions of: 'is it social?' or 'is it technical?' as missing the point, which should be: "is this association stronger or weaker than that one?" (Latour, 1988b :27). In ANT an actor is any human or non-human entity that is able to make its presence *individually* felt (Law, 1987) by the other actors. An actor is made up *only* of its interactions with these other actors, and Law (1992) notes that an actor thus consists of an association of heterogeneous elements constituting a network.

To address the need to treat both human and non-human actors fairly and in the same way, actor-network theory is based upon three principles: agnosticism, generalized symmetry and free association (Callon, 1986b). In summary, ANT attempts impartiality towards all actors in consideration, whether human or non-human, and makes no distinction in approach between the social, the natural and the technological. Actor-network theory, or the 'sociology of translations' (Callon, 1986b; Law, 1992), is concerned with studying the mechanics of power as this occurs through the construction and maintenance of networks made up of both human and non-human actors.

Latour (1986), one of the main proponents of ANT, argues that in an innovation translation model the movement of an innovation through time and space is in the hands of people, each of whom may react to it in different ways. They may accept it, modify it, deflect it, betray it, add to it, appropriate it, or let it drop. The adoption of an innovation comes as a consequence of the actions of everyone in the chain of actors who has anything to do with it. Furthermore, each of these actors shapes the innovation to their own ends and instead of a process of transmission we have a process of continuous transformation (Latour, 1996) where faithful acceptance involving no changes is a rarity. The key to innovation is the creation of a powerful enough consortium of actors to carry it through, and when an innovation fails to be taken up this can be considered to reflect on the inability of those involved to construct the necessary network of alliances amongst the other actors (McMaster, Vidgen, & Wastell, 1997). Getting an innovation accepted calls for strategies aimed at the enrolment of others, and Latour maintains that this is done by 'interesting' others and then getting them to follow our interests, so becoming indispensable to them. This process is facilitated if other possibilities are first blocked off.

Mechanisms of Translation

An actor-network is configured by the enrolment of both human and non-human actors, and this is done by means of a series of negotiations in a process of re-definition in which one set of actors seeks to impose definitions and roles on others. Translation can be regarded as a means of obliging some entity to consent to a 'detour' (Callon, 1986a) that takes it along a path determined by some other entity. Law (1987) uses the term 'heterogeneous engineer' to describe the entity that designs and creates these detours.

The process of translation has four aspects or 'moments' (Callon, 1986b), the first of which is known as *problematisation*. In this stage a group of one or more key actors attempts to define the nature of the problem and the roles of other actors so that these key actors are seen as having the answer, and being indispensable to the solution of the problem. It involves suggesting an equivalence (Law, 1997) between two problems: the one proposed by the enrollers and the other by those being enrolled, and requiring those who wish to solve the problem to accept the solution proposed by the other. In other words, the problem is redefined, or *translated*, in terms of solutions offered by these actors who then attempt to establish themselves as an 'obligatory point of passage' (Callon, 1986b) which must be negotiated as part of its solution. They attempt to persuade the other actors that they all have the same interests and that the answers to their own problems lie in the solutions proposed by the persuaders. To pass through the obligatory passage point the other actors must accept a set of conventions, rules, assumptions and ways of operating laid down by the heterogeneous engineer. If this happens then the formation of a stable network will ultimately result.

The second moment is *interessement* which is a series of processes that attempt to impose the identities and roles defined in the problematisation on the other actors. It means interesting and attracting an entity by coming between it and some other entity. Here the enrollers attempt to lock the other actors into the roles proposed for them and to gradually dissolve existing networks, replacing them by a network created by the enrollers themselves.

If the interessement is successful then the third moment, *enrolment* will follow through a process of coercion, seduction, or consent (Grint & Woolgar, 1997), leading to the establishment of a solid, stable network of alliances. Enrolment, however, involves more than just one set of actors imposing their will on others; it also requires these others to yield.

Finally, *mobilization* occurs as the proposed solution gains wider acceptance and an even larger network of absent entities is created through some actors acting as spokespersons for others. Mobilization requires that these supposed spokespersons are properly able to represent the others and will neither betray them nor be betrayed by them (Callon, 1986b). Of course, not all entities will just willingly consent to the proposed detours, and in understanding the path taken by an innovation it is necessary to examine the resistance offered by the actors it is able to mobilize and those it rejects or that reject it (Latour, 1991).

To define the relationship between themselves many actors make use of intermediaries such as texts, technical artifacts, humans with specific skills, and money (Callon, 1991). These intermediaries then constitute the 'form and substance' of the interactions.

Use of Actor-Network Theory

Examples in the literature show how actor-network theory has been used to investigate the success of a number of technological innovations and, in particular, to describe a number of notable failures. The list that follows gives an indication of the wide range of such studies.

Law (1986; 1987) has used actor-network theory to describe the successful Portuguese exploration down the African coast to trade in India, and the unsuccessful TSR2 project (Law, 1988; Law & Callon, 1988; Law & Callon, 1992) to build a revolutionary military aircraft in Britain. Callon (1986b) has used it to describe the 'domestication' of scallops in St Brieuc Bay, Brittany, and the failure of the Renault car company to develop a successful electric car in France (Callon, 1986a).

Singleton and Michael (1993) have written of the part played by general practitioners in the UK Cervical Screening Program. Grint and Woolgar (1997) have used ANT, and other approaches, to explain the Luddite rebellion and the events surrounding introduction of weaving technology into the United Kingdom in the early nineteenth century.

Latour (1988a) has used actor-network theory to discuss the achievements of Louis Pasteur, some of the processes undertaken by scientists in their research and their laboratories (Latour, 1987), the simultaneous invention of the Kodak camera and the mass market for amateur photography (Latour, 1991), and analysis of the conception and ultimate failure of the revolutionary Parisian public transportation system known as Aramis (Latour, 1996).

One of the things that many people find a little odd about reading accounts making use of actor-network theory is the use of a style of language that has been designed to give agency to the non-human actors. As the language can sometimes be off-putting it could even be considered as a limitation of this approach.

In this paper we have attempted to point out how and why language is used in this way. Another difficulty with ANT is that it has no methods specific to itself, making use of many of the methods, but not the philosophical stance, used in methodologies such as ethnography and case study research.

Data for the study was collected in the form of relevant curriculum documents, and a series of interviews of the academic and support staff involved. Data collection and analysis were undertaken by the authors.

Innovation in the Information Systems Curriculum at Phillip University

While the research reported in this paper is factual, the names of the academic staff involved, and of the university, have been changed. The discussion that follows constitutes an actor-network-informed account of the adoption of Visual Basic in two information systems subjects at Phillip University. It describes how, in the course of attempting to solve a programming problem unconnected with his teaching Fred, an IS academic at the university, discovered Visual Basic, was enrolled by it, and began to make use of it in his teaching. It examines how Visual Basic managed to redefine business programming at Phillip University from the approach offered by character-based procedural programming languages, to its own form of graphical, event-driven programming. We have used an actor-network approach to describe two key moments in this curriculum change:

- how Fred found out about Visual Basic, and
- how he then became convinced that VB's very different style of programming should be adopted in his teaching.

Problems when Programming with Graphics in MS-DOS

Fred recollects that his first experience of Visual Basic was unconnected with his university teaching and occurred when he was working privately with his son George on a small commercial programming project in an MS-DOS environment. The project was for a system to store customer records for a local garage, and was to be programmed in Microsoft QuickBasic. One aspect of the project required the display of a variety of different sized fonts, and some pictures, on the computer screen. But displaying anything other than standard text presented difficulties when using an MS-DOS programming language as each different type and resolution of computer screen required a different MS-DOS screen-driver. The purpose of these screen-drivers was to achieve cooperation between the computer and a specific type of screen and to induce the appropriate parts of the screen to change luminescence according to the wishes of the program running on the computer; to act as an 'intermediary' (Callon, 1991 :134) to get the screen and the program working together.

There was, however, no single standard for computer screens and each of the different type required specific screen-drivers to operate correctly. Each screen type jealously guarded its own standards, and steadfastly refused to listen to commands coming from screen-drivers intended for other types of screen. (The language used in this account reflects that used in actor-network theory and may appear a little strange to anyone unfamiliar with ANT. This expression exemplifies how ANT grants agency to non-human actors.) Writing a program in a language like QuickBasic required locating and loading the correct screen-driver for the type of screen that would be used when the program was run, and persuading it to operate as required. Differences in screen-drivers that the program had to take note of meant that it was very difficult to write a program that would make different fonts, or a picture, look the same on each type of screen.

A programmer needed to *enroll* these screen-drivers, and hence the associated display screens, as allies. Failure to do so meant that cooperation between the program and the display screen would not follow, and the program would not work as intended. One simple solution was for the programmer to specify in advance the screen type and resolution required for use by the program; to require use of a specific-screen driver as an 'obligatory passage point' (Callon, 1986b). The user would then have to obtain a monitor with this type of screen before using the program. There was no great difficulty in 'forcing' the enrolment of specific monitor screens in this way, but it did not solve the general problem of persuading other screen-types and screen-drivers to voluntarily come along and be part of the solution as well. Defining the screen problem in this way; what Callon (1986b) calls *problematisation* would then have required convincing the program's users that, as their problem could be solved by purchasing the 'right' monitor, a general solution that would solve this problem for all screens was of no relevance to them.

Unfortunately, enrolling the screen-drivers as allies in this way was not an easy task as these actors each saw their purpose as providing a solution to the specific problem of mapping text and graphics onto the particular screen type they represented, not as working together to do this for all types of display screen. The single screen-type problematisation offered by the screen-drivers was soon to be changed by Microsoft Windows, but the MS-DOS version of Visual Basic that Fred was about to come across was also able to achieve the same end. It was able to ensure the cooperation of the screen-drivers in providing consistent output for each type of display screen by enrolling these actors and making them fall into line and do what they were told by the programmer.

Fred discovered that Visual Basic for MS-DOS had a single set of graphics-drivers that worked as 'calling routines' to the operating system rather than going directly to the screen devices. Visual Basic for MS-DOS had enrolled these screen-drivers by incorporating them as a part of its own actor-network, and when Visual Basic for MS-DOS was installed on the hard disk it automatically installed all the device-drivers ready for use with a variety of different display screens. These drivers had been convinced of the advantages of working with it, and been enrolled into a new network: Visual Basic for MS-DOS, that contained both a programming language and other programming elements including device-drivers that constituted an integrated development environment. It would now no longer be necessary for programmers to make all these associations and connections themselves.

This problematisation of programming was enough inducement to Fred to give Visual Basic a try. VB's redefinition of programming in which a programming language was granted the ability to call graphics-drivers through the MS-DOS operating system *interested* (Latour, 1986) both George and Fred. VB's problematisation of graphics programming appeared to them to fit well with their commercial programming problem in the way that they saw it. The solution suggested by VB for MS-DOS offered considerable inducement or, as Callon (1986b) would say, *interessement* for Fred to move away from QuickBasic and to adopt Visual Basic in its place. It was, however, not every aspect of VB that had been significant here, just its ability to work well with graphics screens. This *translation* of VB as a graphics programming language (Callon, 1986b) had successfully enrolled Fred to its view of programming.

A consequence, unexpected by Fred who had always been reluctant to use Microsoft Windows, was that he soon began to like the visual aspects of VB and the different type of programming style it represented. While George's main interest in Visual Basic had been in solving the specific graphics display problem, Fred began to see another side to it that might be relevant to his teaching, leading to the formation of a 'Fred + VB for MS-DOS' hybrid (Latour, 1993). Fred liked the fact that VB allowed the programmer to put a program together in a short time and one, what is more, that looked really good on the screen.

VB problematised (Callon, 1986b) programming tasks quite differently to other languages Fred had been used to, by using drag and drop controls and event-driven code in place of a character-based environment and the use of procedural code. Although Fred had always enjoyed the challenge represented by programming, he found that using VB was 'more fun' than normal programming.

Exploration: Teaching Experiments with Visual Basic

In what can be described as the second key moment of this account, Fred then set out to try to interest others in using Visual Basic. He began with his students, approaching this task cautiously as, although he had been enthused by VB and saw great advantages in its use, he still had to convince himself that it was teachable, and right for his students.

By offering a new and, what seemed to him, better solution to his technical problem with graphics screens VB had enrolled Fred to its view of programming and he now had to work out what this meant to his teaching. Choice of VB as a solution to the technical problem had created an education dilemma for Fred: should he try to introduce Visual Basic into his course, and if so, how? Fred was impressed with VB's consistent visual environment and the speed with which VB applications could be developed. He was, however, not sufficiently sure of how the students would take to it to propose the creation of a new subject to teach Visual Basic. He was also not sure whether he could develop enough materials for its use and so decided instead to try it out first in existing subjects.

At the time that his discovery of Visual Basic for MS-DOS was going on, Fred's teaching at Phillip involved programming in Cobol on the VAX and Pascal in MS-DOS. He remarked that in writing a program in either of these languages, everything you wanted to add to a form to display to the user had to be instigated by a separate line of program code.

Fred described how he first tried out Visual Basic as a screen prototyping tool in the subject 'Business Information Systems-A'. He did this not just by accepting VB as Microsoft had designed it, but by *translating* (Callon, 1986b) it from a 'programming language and visual programming environment' into a 'screen prototyping tool'. The prototyping topic in this subject had always been hard to teach as, without suitable prototyping tools to use in the student labs, it could not be handled practically. Fred assigned Visual Basic a new role in the demonstration of screen prototyping, and VB accepted this new role. Reassignment of Visual Basic's role was a good deal easier than it sounds and just meant completing only the first step of program development in VB by using its visual interface to create the screen that would be seen by the user at runtime, but not proceeding to the stage of adding program code. This translation meant leaving out most of VB's object and event-driven programming features and introducing just its visual interface. It was a translation that reduced the scope of Visual Basic to something that was appropriate to this subject, making it possible to use VB in this role.

Fred indicated that he thought that any easy way to convert this subject from pure theory into something more practical had to be an improvement. The slight change in the problem definition that allowed this was seen as highly desirable both by the teacher and the students. The translation of VB to hide its programming features and leave just its visual design interface enabled it to be of use here. So where CASE tools and the VAX screen painter had been unable to offer a redefinition of 'Business Information Systems-A' that would allow their enrolment, Visual Basic was able to do so.

One difficulty though was that Fred was then far from expert in using or teaching VB. Following the old adage that 'the best way to learn something is to teach it', Fred's own learning about the capabilities and limitations of Visual Basic, and how it could be used in business programming, proceeded in parallel with his experiments in teaching with it (Latour, 1987). Fred solved his own problems in dealing with the new design and programming paradigms required by Visual Basic by working through them with his students. The hybrid (Latour, 1993) had now enlarged to become 'Fred + VB for MS-DOS + students'.

Before being adopted in this subject, however, Visual Basic 'the programming language', had undergone a *translation* to become Visual Basic 'the interface design and prototyping tool'; with the hybrid of Fred and the students attached. Without this translation Visual Basic could not have been adopted as many of its attributes were not at all relevant for use here and could get in the way. What Fred required was a simplified 'cut-down' version of Visual Basic that removed all of its programming and object-based features and left only its abilities in user-interface design. This translation allowed VB to enter the curriculum.

The next semester Fred tried out the Windows version of Visual Basic in another subject: 'Operating Systems Programming'. Previously this subject had concentrated on programming within a Unix environment with students using Unix commands and writing scripts for operating systems procedures. With the growing importance of Microsoft Windows, Fred justified VB's inclusion here to write operating system extensions for Windows instead of Unix.

After using it for a while Fred noted Visual Basic's low threshold for relative beginners; students with little or no experience of programming could soon learn to make enough use of VB to produce impressive looking programs. Fred had modified, by another small translation, the definition of 'Operating Systems Programming' from a subject originally intended to examine the Unix operating system, to one that also looked at Microsoft Windows through the use of VB. Visual Basic was adopted after it underwent a different translation, this time to become Visual Basic, the 'language for Windows operating systems programming'. In this, and the previous instance of prototyping with Visual Basic, we begin to see the emergence of two new 'Fred + students + VB for MS-DOS' hybrids (Latour, 1993) that had been mobilized to speak for and advocate the use of appropriate aspects of Visual Basic to solve the problems of prototyping and operating systems programming.

Innovation Translation as a Means of Understanding Change

There were now two different *translations* of Visual Basic in Phillip University's curriculum, rather than Visual Basic itself as it might have been envisaged by Microsoft. Neither of these represented all aspects of Visual Basic; they were translations of the original that used only certain aspects of it and so enabled its adoption in the curriculum. It could appropriately be said that it was not Visual Basic as such that was adopted, but rather 'VB the screen prototyping tool' and 'VB the language for Windows operating systems programming'.

Innovation translation offers an approach to theorizing innovation that has the advantage of being able to single out and concentrate on those aspects of an innovation that really do influence whether or not it is adopted in *each specific case*, rather than just globally. While other approaches, such as innovation diffusion, need to consider supposed innate properties of an innovation that are thought to determine its adoption, innovation translation does not have to resort to such essentialist notions. An innovation translation approach, informed by actor-network theory, avoids the need to consider the social and the technical, and thus human and non-human actors, in different ways. This makes innovation translation very useful in modeling the progress of an information systems curriculum innovation.

Actor-network theory avoids use of any notions of cause and effect, choosing instead to highlight how the various actors (both human and non-human) interact, and how these interactions may lead to the formations of stable networks. It would thus be inappropriate to look too hard at how this account might point to how other examples of IS curriculum change may occur, except to point out how IS curriculum change will typically involve a series of complex interactions between human and non-human actors. The authors contend that ANT can be applied most successfully to identifying and illustrating these complex interactions.

References

Brey, P. (1997). Philosophy of Technology meets Social Constructivism. *Society of Philosophy & Technology, 2*(3-4).

Callon, M. (1986a). The Sociology of an Actor-Network: The Case of the Electric Vehicle. In Callon, M., Law, J., & Rip, A. (Eds.), *Mapping the Dynamics of Science and Technology* (pp. 19-34). London: Macmillan Press.

Callon, M. (1986b). Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. In Law, J. (Ed.), *Power, Action & Belief. A New Sociology of Knowledge?* (pp. 196-229). London: Routledge & Kegan Paul.

Callon, M. (1991). Techno-Economic Networks and Irreversibility. In Law, J. (Ed.), *A sociology of monsters. Essays on power, technology and domination* (pp. 132-164). London: Routledge.

Callon, M. (1999). Actor-Network Theory - The Market Test. In Law, J. & Hassard, J. (Eds.), *Actor Network Theory and After* (pp. 181-195). Oxford: Blackwell Publishers.

Grint, K., & Woolgar, S. (1997). *The Machine at Work - Technology, Work and Organisation*. Cambridge: Polity Press.

Havelock, R. (1971). *Planning for Innovation Through Dissemination and Utilization of Knowledge* (Paper). Ann Arbor: Center for Research on Utilization of Scientific Knowledge, Institute for Social Research, University of Michigan.

Kaplan, B. (1991). Models of Change and Information Systems Research. In Nissen, H.-E., Klein, H. K., & Hirschheim, R. (Eds.), *Information Systems Research: Contemporary Approaches and Emergent Traditions* (pp. 593-611). Amsterdam: Elsevier Science Publishers.

Latour, B. (1986). The Powers of Association. In Law, J. (Ed.), *Power, Action and Belief. A new sociology of knowledge? Sociological Review monograph 32* (pp. 264-280). London: Routledge & Kegan Paul.

Latour, B. (1987). *Science in Action: How to Follow Engineers and Scientists Through Society*. Milton Keynes: Open University Press.

Latour, B. (1988a). *The Pasteurization of France* (Alan Sheridan and John Law, Trans.). Cambridge, Ma.: Harvard University Press.

Latour, B. (1988b). The Prince for Machines as well as for Machinations. In Elliott, B. (Ed.), *Technology and Social Process* (pp. 20-43). Edinburgh: Edinburgh University Press.

Latour, B. (1991). Technology is society made durable. In Law, J. (Ed.), *A Sociology of Monsters. Essays on Power, Technology and Domination* (pp. 103-131). London: Routledge.

Latour, B. (1993). *We have never been modern* (Catherine Porter, Trans.). Hemel Hempstead: Harvester Wheatsheaf.

Latour, B. (1996). *Aramis or the Love of Technology*. Cambridge, Ma: Harvard University Press.

Law, J. (1986). On the methods of long distance control: vessels, navigation and the Portuguese route to India. In Law, J. (Ed.), *Power, Action and Belief: A New Sociology of Knowledge?* (pp. 234-263). London: Routledge & Kegan Paul.

Law, J. (1987). Technology and Heterogeneous Engineering: The Case of Portuguese Expansion. In Bijker, W. E., Hughes, T. P., & Pinch, T. J. (Eds.), *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology* (pp. 111-134). Cambridge, Ma: MIT Press.

Law, J. (1988). The Anatomy of a Socio-technical Struggle: the Design of the TSR2. In Elliott, B. (Ed.), *Technology and Social Process* (pp. 44-69). Edinburgh: Edinburgh University Press.

Law, J. (1992). Notes on the Theory of the Actor-Network: Ordering, Strategy and Heterogeneity. *Systems Practice, 5*(4), 379-393.

Law, J. (1997). Topology and the Naming of Complexity (Draft), *Actor Network and After Workshop*. http://www.lancaster.ac.uk/sociology/stslaw3.html, 31 July 1997: Department of Sociology, Lancaster University.

Law, J., & Callon, M. (1988). Engineering and Sociology in a Military Aircraft Project: A Network Analysis of Technological Change. *Social Problems, 35*(3), 284-297.

Law, J., & Callon, M. (1992). The Life and Death of an Aircraft: A Network Analysis of Technical Change. In Bijker, W. & Law, J. (Eds.), *Shaping Technology/Building Society: Studies in Sociological Change* (pp. 21-52). Cambridge, Ma.: MIT Press.

McMaster, T., Vidgen, R. T., & Wastell, D. G. (1997, 9-12 August, 1997). *Towards an Understanding of Technology in Transition. Two Conflicting Theories.* Paper presented at the Information Systems Research in Scandinavia, IRIS20 Conference, Hanko, Norway.

Nordvall, R. C. (1982). *The process of change in higher education institutions* (ERIC/AAHE Research Report 7). Washington DC: American Association for Higher Education.

Rogers, E. M. (1995). *Diffusion of Innovations.* (4th ed.) New York: The Free Press.

Singleton, V., & Michael, M. (1993). Actor-Networks and Ambivalence: General Practitioners in the UK Cervical Screening Programme. *Social Studies of Science, 23,* 227-264.

Tatnall, A. (2000). *Innovation and Change in the Information Systems Curriculum of an Australian University: a Socio-Technical Perspective.* Central Queensland University, Rockhampton.

Tatnall, A., & Gilding, A. (1999). *Actor-Network Theory and Information Systems Research.* Paper presented at the 10th Australasian Conference on Information Systems (ACIS), Wellington.

Biographies

Arthur Tatnall is a senior lecturer in the School of Information Systems at Victoria University in Melbourne, Australia. His research interests include innovation and change management, information systems curriculum development, Visual Basic programming, project management and electronic commerce.

Bill Davey is a senior lecturer in the School of Business Information Technology at RMIT University, Melbourne, Australia. His research interests involve methodologies for systems analysis and systems development, information systems curriculum, and Visual Basic programming.

Arthur and Bill have co-operated and worked together on many occasions. They have co-authored a number of papers, book chapters, and textbooks relating to information systems and IS curriculum.