2001

Negotiating the Chasm in System Development: Some Advice from ANT

Jim Underwood
University of Technology, Sydney, jim@it.uts.edu.au

Negotiating the Chasm in System Development: Some Advice from ANT

Jim Underwood

Department of Information Systems
University of Technology, Sydney, Australia
jim@it.uts.edu.au

Abstract

Apparent failures in IS development projects are often attributed to mistranslations of requirements as the system is developed, to language incompatibility between users and technical experts. Many development methodologies claim to overcome this misunderstanding through forcing acceptance of a common language or through formalising the translation process, but we claim that this merely covers up the problem. Based on constructivist models of communication and a case study we advocate techniques which will keep misunderstandings visible thoughout the life of an information system.

Keywords

IS development strategies, participative design, actor-network theory

SEARCHING FOR A SEAMLESS DESIGN APPROACH

Between the idea
And the reality
Between the motion
And the action
Falls the Shadow (Eliot, 1925)

The theory of IS development has been haunted by the chasm between idea and reality, by the failure of the systems we construct to live up to our imaginings. (Lyytinen and Hirschheim, 1987) This shadow may be more theoretical than practical. While academics, managers and accountants believe that many IS projects, if not total failures, are at least very expensive mistakes, the developers themselves seem to believe that they are doing as well as could be expected and are producing many highly valuable systems. For managers the gap between idea and reality is caused by inappropriate culture or lack of commitment on the part of the developers; for theoreticians the cause is a failure to use or properly adhere to an appropriate methodology.

Traditional IS developments follow a "life-cycle" approach, prescribing a number of distinct development phases to be followed more or less sequentially. A typical sequence of development phases is shown in Figure 1. A major source of dissatisfaction, the chasm between idea and reality, becomes apparent when the behaviour of the installed system in the real organisation does not match the expectations remembered from the time of requirements development. Logically this implies that some mistake has been made, some mistranslation has occurred, in analysis, in installation or in the passage from one development phase to the next. To avoid mistranslation at the installation stage the usual recommendations are training and planning. For the preceding steps the solution is often seen to lie with the proper choice of model used by the methodology. To facilitate the translation at point "a" of Figure 1 some methodologies claim to use models that correspond to "natural ways of thinking". The movement between development stages is facilitated by having a "seamless" methodology which applies the same model, in more detail, at each stage (Jacobson, 1992:42), or by having robust algorithms to translate the models of one stage to those of the next (Hawryszkiewycz, 1998). Uniform or reliably translatable models can promote either business or technology as the dominant discourse. Nilsson et al (1999) have created a methodology which attempts to translate the language of strategic planning through all stages of system development, while object oriented approaches attempt to have users "think in objects" from the beginning.

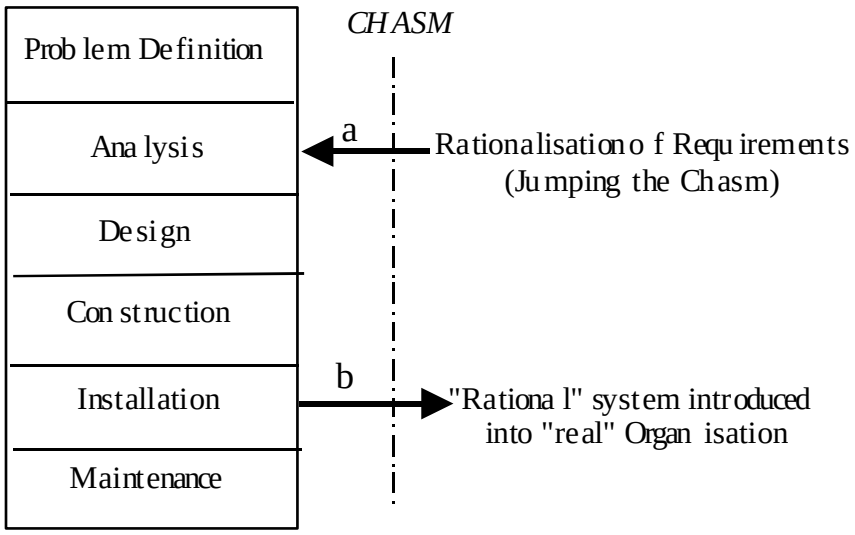

Figure 1: - Traditional Development Life Cycle

One difficulty with this type of approach is illustrated by the following thought experiment. Suppose we simplified the translation in step "a" by recording the results of analysis in natural language, a natural way of thinking for most potential users. It would then be possible to create algorithms, based on some formal model of the organisation, to translate this natural language into code to control the computer component of an information system (Rist, 1994). The problem is, would this translation correspond to what the users really meant. Would they trust the translation? A second difficulty is that a high percentage of system failures are attributed tochanges in organisational needs during the project (Fitzgerald et al, 1998). Thus uncertainty can occur in two pathways, through the processes of the development methodology and within the user organisation, external to the project. Crozier (1964) tells us that control of uncertainty is an important contributor to organisational power, so it is understandable that both technical experts and users will attempt to claim their own areas of uncertainty (expert judgement, creative freedom) while insisting that the other "stick to the specification".

A major contributor to the above difficulties is the notion that system requirements can be frozen in time at point "a". Except for very short projects or quite static organisations this is clearly impractical. Hauffe (1998) sees design as " a process in which the form of a product comes into existence alongside the determination of its function", that is the requirements are developed concurrently with the product. This is the basis of an alternative approach to system development, prototyping, which has been available and used for several decades and is illustrated in Figure 2. Here the stages of development are less distinct and the shadow between idea and reality is always visible to both developers and users. While even the more democratic developers sometimes try to impose limiting models on their discussion with users (Trigg et al, 1991), this shadow is always a site of ambiguity where negotiations between competing discourses take place. In a recent research project, our aim was to find a model of how these negotiations proceed and how the way they are conducted can affect the outcome of a development project.

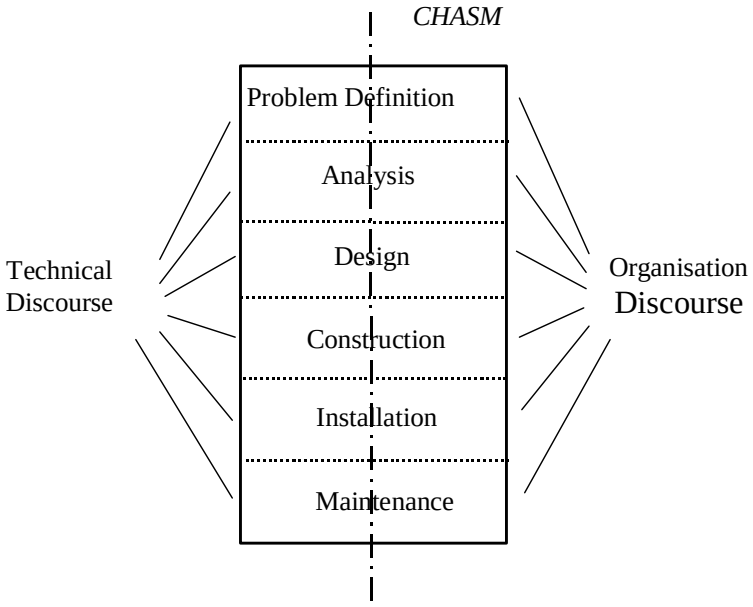

Figure 2: - Prototyping

USING ANT TO UNDERSTAND SYSTEM NEGOTIATION

The recommendations in this paper arose from a case study of an internet based flexible delivery project undertaken at an Australian university (not the author's university). This was a one year pilot project on a fixed budget and was a user developed system, with a diverse community of potential users. Although the software envisioned in early discussions was not produced, the majority in the user community saw the project as a success. The original intention of the case study research was to find a suitable semiotic model to describe the development or otherwise of shared meanings among the project participants. The progress of this research, based on an analysis of design documents, e-mail discussions, minutes of meetings and interviews is described in Underwood (1998).

The model finally chosen was a combination of Foucault's discourse theory and actor-network theory (ANT). We have mentioned "discourse" informally above, and in Figure 2. For Foucault (1972), a discourse is a network of allowable potential statements (and their sanctioning institutions) which forms the basis of serious talk in some discipline, such as accounting or computer science. The meaning of a statement is interpreted within a particular discourse. ANT (Latour, 1992) is a type of stakeholder analysis (Mitroff, 1983) where the stakeholders may be other than the usual people and organisations and the stakes may be reinterpreted as negotiations progress. In ANT, however, the stakeholders (*actors*) are seen as internal to the network and so have reciprocal influence on each other as well as on "the project" or any imagined manager. Influence is exerted by *inscribing* other actors with *scripts* that will cause them to assist our project. A plan for inscribing all necessary actors with the appropriate scripts is called a *program* and will of course be challenged by *anti-programs*. Ideally an actor, be they a project, process, machine, idea or person, would like to be seen as a *blackbox* whose behaviour is taken for granted and is not amenable to rescripting. Others, including researchers, wish to *de-scribe* actors to understand their motives and possibly influence their behaviour. In our research we used Foucault's discourse theory to access the content of scripts and to understand how the same script was interpreted differently by different actors or by the same actor at different times. This is discussed in some detail in Underwood (2001). In the current paper we concentrate more on the details of negotiation across the chasm.

As scripts pass from one actor to another they are translated and change their meaning. Whether this change is seen as a faithful translation or as a betrayal may change over time as the implications of the translation work their way through the network.

> ... there is always a difference between what a speaker means and what the speaker's words mean. Moreover, signifiers produce signification (meaning) and that meaning is often constructed retroactively. (Sarup, 1992:57)

In our case study we found two levels of scripting (figure 3). Espoused scripts such as "support flexible delivery" were apparently subscribed to by new recruits while still retaining the script's identity. We suppose (though this is hard to test) that these were "anchored" or personalised by various actors through some process of

internal translation, possibly through reference to a preferred discourse. These then emerge as (different) scripts-in-use, analogous to Argyris and Schon's (1978) theories-in-use. The translation/betrayal of these scripts-in-use seems to happen externally in the network rather than within the actors. Thus we have conceptually two networks, one of espoused scripts and one of scripts-in-use. It is not clear how these two levels interact, but it is certain that neither can be understood independently.

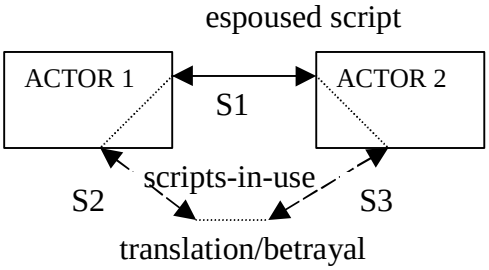

Figure 3 - Two levels of network

When these translations are regarded as betrayals, the internal translation is seen as failing through perversity or ignorance of actors, the external through lack of a common language. Soft and hard systems development methodologies respectively claim to deal with one or the other of these sources of mistranslation. Theoretically, unless we specify some super actor whose discourse is dominant (such as the project manager), there can be no distinction between translation and betrayal. The human participants in our case study were not particularly concerned about this possibility of mistranslation, generally assuming that it occurred but not showing much interest in where or how ("that's politics!"). These betrayals and the consequent ambiguity can in fact be used to maintain stability in the project alliance. In the case study a major part of the non-technical activity was devoted to having people "get with the program" - that is to the phases of *interessement* and *enrolment* (Callon, 1986), where actors are first attracted to the project, and then retained through the cultivation of alliances and common interests. Contrary to Argyris's (1978) apparent advice, a mixture of both artfulness and openness is required for success. The art of promoting a project involves both finding the vague concepts on which we can all agree and recognising the currently conflicting specific scripts which can be linked through mistranslation. One of the keys to success in the case study project was to separate these two modes of scripting. Many times at project working group meetings there were attempts to force agreement on the meaning of particular concepts, goals or design elements. Whenever it became apparent that this was difficult the matter was deferred. This meant that a script was not translated at all (in which case it dropped out of the network - for example "seek commercial sponsorship for the web site") or a translation was allowed to develop without scrutiny (eg "support flexible delivery" to "put lecture notes on the web"). This unremarked translation/betrayal was crucial in maintaining "commitment" to the project.

ADVICE FOR PROJECT MANAGERS

Participants in IS development projects, and particularly project managers, usually concentrate on one or other of the paths introduced in Figure 3. They either take a Machiavellian view and promote superficial agreement and high sounding concepts while secretly working to their own goals, or they insist on all players subscribing to detailed design specifications expressed in the language of some dominant discourse. They attempt to exert either covert or overt control. It might be thought that to deal with both paths a participant needs to strive for both types of control, but our experience with the case study suggests that successful project management is a matter of facilitation rather than control. In Latour's version of ANT we can't place ourselves outside the network. The project manager is just one more actor, who can promote programs through inscription of other actors but cannot expect to control the entire network. Further, openly with prototyping and unnoticed in life-cycle methodologies, the requirements for the system are constructed gradually along with the system, so there are no fixed goals against which to measure control. To help the project manager in such a situation we suggest a number of scripts which they might follow.

Don't be Afraid of Politics

For technically-oriented or inexperienced analysts and project managers, politics is usually regarded as an intrusion, a distraction from the real work of the project. ANT provides a political model which can include the interests of ambitious individuals, bureaucratic departments, technical artifacts, investors and the nominated developers themselves. If politics are imagined as other, their effects on the project and operational information system will be external disturbances, outside our control. If political scripts are routinely included in our network they can, as with any other scripts, be more or less successfully translated to support our program.

Keep Project Boundaries Broader and Less Technical

Information systems are usually defined as comprising people, processes, hardware and software, but many development methodologies concentrate on the hardware and software. An actor-network view of an information system can be similar to Checkland's (1972) human activity system, with an emphasis on the activities and interests of the human actors. The question of which actors should be legitimately included in a project network is discussed by McMaster et al (1998). Our model, through concepts, allows scripts of non-actors (those outside the network) to be included as well, and we suggest project managers become comfortable with the fact that the network will evolve, sometimes becoming more focussed, sometimes growing, as the project progresses.

Avoid Hierarchies of Project Definition

This continual evolution of the project undermines any techniques for hierarchical definition such as structured analysis, encapsulation (Jacobson, 1992:48) or "sign-offs". Traditional systems analysis recommends the production of a simple top level model, the components of which are specifications for the behaviour of systems at the next lower level, with this process iterated many times for a complex system. In an actor network there are actors, scripts and programs, but within these there are no hierarchical levels. The apparently least significant factor may rise to the highest importance at any moment.

> … by following the movement allowed by ANT, we are never led to study the social order, in a displacement that would allow an observer to zoom from the global to the local and back. In the social domain there is no change of scale. It is so to speak always flat and folded ... (Latour, 1999:18)

Explicitly Discuss the Interests of Possible Technical Actors

Despite our plea for greater emphasis on non-technical aspects of information systems, the technical should not be ignored or relegated to a "lower level" analysis. Scripts relating to technical aspects of possible solutions should be introduced to the network as soon as they are found. Scripts such as "always use IBM products", "the maximum modem speed is 57K" or "don't use Mac servers" are neither unquestionable constraints nor unexpected difficulties that arise in the later stages of implementation. They are on a par with "keep cost below $100,000", "provide online feedback on student assignments" and "avoid gender bias", and may be involved in facilitating or blocking alliances at any stage of IS development.

Keep Track of Important Scripts

Hughes et al (1995) kept actors' views of roles, rules and work flows (as discovered in interviews) in a hypertext database so that designers could have access to the originals, uncontaminated by any process of requirements definition. Similarly, we recommend what could be called a "script audit", where significant scripts are recorded and followed. It is not expected that all scripts will be satisfied by the project, but if a significant script is dropped or drastically changed this should be noted. The function of the script audit is to detect those scripts which have simply been forgotten, since they will probably reappear at an inconvenient time.

Continue De-Scribing

The aim of any project is to be blackboxed, to become part of the environment, something which is taken for granted. In that state the scripts which the project is following are invisible and the project is transformed into a politically neutral artifact. Those actors who have succeeded in having their programs embodied in the artifact (Winner, 1977) have "won", and the others are biding their time, expressing their disagreement through passivity, sabotage, plotting revenge or de-scribing the artifact as a tool of the oppressors. To prevent this situation, the project sponsors (and project manager) need to be continually de-scribing, keeping open decisions about what scripts the project supports. Ideally, all actors should practice this de-scribing, even those who seem to be "winning" at the moment. And this de-scribing will continue when the project has been "completed" to become an operational information system, when de-scription becomes institutionalised in "support", "maintenance" and "enhancement".

Avoid Linguistic Dominance

IS developers need to resist imposing their language on their clients. Those who have succumbed to the temptation include numerous advocates of structured analysis and object oriented design, and even Stafford Beer (1975) with claims that "everything is an information system". Other discourses also systematically attempt to dominate IS development, discourses such as project management ("define testable deliverables"), cost management ("on time and within budget"), marketing and strategic business management ("align IT with business strategy"). Through language dominance actors hope (consciously or otherwise) to pre-empt decisions which might otherwise be made during the development or even the operation of the information system.

The converse of dominance by a home discourse of one (or some) of the actors, is dominance by the project's local language (Westrup, 1999). Words, phrases and other signs such as diagrams come to have an agreed meaning within the project network. Should the actors become too comfortable with this local language they will betray their home discourses, resulting in eventual rejection of the system by those whom they represent. An early symptom of this is when outsiders perceive their colleagues inside the project network as speaking incomprehensible jargon.

Patch Up (But Don't Cover Up) Differences

The previous script advises us at Foucault's macro level of language or discourse. Here we consider what can be done at the more detailed level of individual translations. The model of (mis)understanding introduced in Figure 3 shows "patching up" mechanisms in place, both the superficial consensus of espoused scripts and the mistranslations of scripts-in-use. Amongst those actors who care for the project there is a strong desire for this patching up to succeed, a desire closely related to that noted in ethnomethodological conversation analysis.

> "The big question is not whether actors understand each other or not. The fact is that they do understand each other, that they *will* understand each other, but the catch is they will understand each other regardless of how they would be understood." [Garfinkel The Perception of the Other: a Study in Social Order unpublished PhD, Harvard, 1952:367] (quoted in Heritage, 1984: 119)

The construction of espoused scripts that gain consensus, the enrolment of actors and the procurement of carers for the project co-produce and support each other. If Garfinkel is right and the will to understand is deeply ingrained, then facilitating the construction of a superficial consensus may be the first step to project success. These agreed espoused scripts appear in formal documentation and meetings. For actual work to proceed, however, the second mode of patching up, translation of scripts-in-use, must progress, and this occurs more informally. According to conversation analysis this "alignment" is a matter of perception or assumption. The first actor makes a statement, the second "pretends" that they understand and produces a response; the first actor takes this response as a sign that they have indeed been understood. The process is repeated until one of the actors becomes so uncomfortable with the pretence of understanding that they disrupt the normal flow of conversation and take the risk of undermining the whole appearance of consensus.

Remenyi (1999:18) gives a good overview of this process of continual translation, questioning and accommodation.

> A process of continuous and dynamic evaluation and debate between knowledgeable stakeholders recognising the need for shared values, individual autonomy and ambiguity, provides the best chance for information systems optimisation.

CONCLUSION - COULD IT WORK?

The above scripts put anyone who is designated or assumed to be project manager in a difficult position. While they, above all other actors, are supposed to be committed to the success of the project, its reaching the status of a completed black box, the suggested scripts are designed to undermine this end. The simplest, most important and most difficult script recommended for all actors, but particularly project managers is "let go". Managing a project can be compared to teaching students or bringing up children. We have some ideas about what we hope to achieve, some knowledge and experience, plenty of advice to give and a few techniques (of doubtful efficacy) for influencing behaviour. We feel responsible, we care deeply about the outcome, but we should be neither surprised nor disappointed when the reality turns out quite differently from anything we might have expected. In this sense all projects are "research projects". Even when the hardware and software to be used are "tried and tested" they are being introduced to a particular and unique social situation. Loving the project means supporting it and allowing it the freedom to grow. If any actors (including the project manager) insist on making the project their creature then indeed "outside everything is cast in stone" and the project will not thrive. (Latour, 1996)

Given the usual environment of information systems development, our advice to relinquish control, nurture opposition and rely on the good will of other actors are unlikely to be received with enthusiasm. As well as the fear (and sometimes impossibility) of defying budgets and timelines and of treating top management as "just another actor", there is the risk that, the more successful the project manager is in their role of facilitating translation throughout the network, the less noticed their work will be; for a full-time project manager, disappearing from notice might not be a good career move. We can only conclude with the suggestion:

> The purpose of getting power is to be able to give it away. (Aneurin Bevan, quoted in Foot, 1962)

REFERENCES

Argyris, Chris and Schon, Donald (1978) *Organizational Learning* Addison-Wesley, Reading MA.

Beer, Stafford (1975) *Platform for Change* Wiley, London.

Callon, Michel (1986) "Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of Saint Brieuc Bay" in Law, John (ed) (1986) *Power, Action and Belief: a new Sociology of Knowledge?* Sociological Review Monograph, Routledge and Kegan Paul, London , pp196-233.

Checkland, Peter B. (1972) "Towards a System-Based Methodology for Real-world Problem Solving" *Journal of Systems Engineering 3,2.*

Crozier, Michel (1964) *The Bureaucratic Phenomenon* University of Chicago.

Eliot, T. S. (1925) "The Hollow Men" *Selected Poems* Faber and Faber, London (pub 1954) s5.

Fitzgerald, Guy; Phillipides, Antonios; Probert, Steve (1998) "Information Systems Development, Maintenance and Flexibility: Preliminary Findings from a UK Survey" in *Proceedings of the 6th European Conference on Information Systems* Aix-en-Provence, Jun 4-6, pp1600-1607.

Foot, Michael (1962) *Aneurin Bevan, v1.*

Foucault, Michel (1972) *The Archaeology of Knowledge* Tavistock, London.

Hauffe, Thomas (1998)*Design: A Concise History* Laurence King, London.

Hawryszkiewycz, I. T. (1998) *Introduction to Systems Analysis and Design 4th edn* Prentice Hall, Sydney.

Heritage, John (1984) *Garfinkel and Ethnomethodology* Polity Press, Cambridge.

Jacobson, Ivar (1992) *Object-Oriented Software Engineering* Addison-Wesley, Wokingham.

Latour, Bruno (1992) "Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts" in Bijker, Wiebe E. and Law, John (eds) *Shaping Technology / Building Society* MIT Press, Cambridge.

Latour, Bruno (1996) *ARAMIS or The Love of Technology* Harvard University Press, Cambridge MA.

Latour, Bruno (1999) "On recalling ANT" in Law, John and Hassard, John (eds) *Actor Network Theory and After* Blackwell, Oxford.

Lyytinen, K and Hirschheim, R. (1987) "Information Systems Failures: a Survey and Classification of the Empirical Literature" *Oxford Surveys in Information Technology 4* pp257-309.

McMaster, Tom; Vidgen, Richard T.; Wastell, David G. "Networks of Association and Due Process in IS Development" in Larsen, Tor J.; Levine, Linda; de Gross, Janice I. (eds) *Information Systems: Current Issues and Future Changes: Proceedings of IFIP WG8.2 & WG8.6 Joint Working Conference* Helsinki, December 10-13, pp341-357.

Mitroff, Ian I. (1983) *Stakeholders of the Organizational Mind* Jossey-Bass, San Francisco.

Nilsson, Anders G.; Tolis, Christofer; Nellborn, Christer (eds) (1999) *Perspectives on Business Modelling* Springer, Berlin.

Remenyi, Dan; White, Terry; Sherwood-Smith, Michael (1999) "Language and a post-modern approach to information systems" *International Journal of Information Management 19,1* February, pp 17-32.

Rist, R. S. (1994) "A cognitive model of system development" *First Australian Seminar on Modelling and Improving System Development* Melbourne, pp. 67-83.

Sarup, Madan (1992) *Jacques Lacan* Harvester, New York.

Satzinger, John W.; Jackson, Robert B.; Burd, Stephen D. (2000) *Systems Analysis and Design in a Changing World* Thomson Learning, Cambridge MA.

Trigg, Randall H.; Bødker, Susanne; Grønbaek, Kaj (1991) "Open-ended Interaction in Cooperative Prototyping" *Scandinavian Journal of Information Systems 3* pp63-86.

Underwood, Jim (2001) " Translation, Betrayal and Ambiguity in IS Development" *Proceedings of IFIP WG8.1 Working Conference* Montreal, July 23-25.

Underwood, Jim (1998) "Not Another Methodology: what ANT tells us about Systems Development" in Wood-Harper, A.T.; Jayaratna, Nimal; Wood, J.R.G. (eds) (1998) *Methodologies for Developing and managing Emerging Technology based Information Systems* Springer, London.

Westrup, Christopher (1999) "Knowledge, Legitimacy and Progress? Requirements as Inscriptions in Information Systems Development" *Information Systems Journal 9,1* January, pp35-54.

Winner, Langdon (1977) *Autonomous Technology* MIT Press, Cambridge MA.

COPYRIGHT