Towards an Understanding of FLOSS: Infrastructures, Materiality and the Digital Business Ecosystem

Mary L. Darking and Edgar A. Whitley

In this paper we present empirical work detailing the engagement practices of a large FLOSS project, the Digital Business Ecosystem (DBE). In common with many other FLOSS projects, the DBE project focused on the development of infrastructural software components. Infrastructures and FLOSS software exhibit multiplicity: as objects they both change and stay the same. Whilst the implications of multiplicity with respect to infrastructure have been well-documented, with respect to FLOSS, they remain under-explored. Through examining how the DBE engaged new participants we were able to explore the nature of the FLOSS software object by asking the implied question: engagement with what? We draw on recent analysis by Law and Singleton to show how the innovative yet non-existent potentiality of the DBE was as significant to engagement as its steadily growing codebase. We argue that acknowledging the materiality and immateriality of the FLOSS software object has important consequences for management of, and engagement with, FLOSS projects.

Keywords: FLOSS, infrastructures, innovation, materiality, engagement

The profile of free/libre open source software (henceforth FLOSS) has risen dramatically in recent years, both commercially and in terms of academic research on the subject. To some, FLOSS is the basis for a new way of organizing economic activities that could replace the traditional notions of market and hierarchy (Benkler, 2002); to others it is the basis for a new form of social order, bringing to the fore a (new) set of values (Himanen, 2001). At a more mundane level, the development of FLOSS is claimed to result in better quality code than traditional software development practices (e.g., Mockus, Fielding and Herbsleb, 2002; Stamelos, Angelis, Oikonomou & Bleris, 2002) or at least better than one might expect from such a loosely structured process.

As a result, FLOSS has been studied from a variety of perspectives (e.g., Management Science, 2006). Amongst the most common are computing (how can this 'undisciplined' approach to systems development be organised and can it deliver more successful systems implementations than traditional methods?), law (what legal protections exist for FLOSS and can they be applied beyond the software field?) and political economy (how can a mechanism that does not

reward individual effort and is, effectively, a public good continue to produce high quality software?). For a recent review of these main streams of research, see von Krogh & Spaeth (2007).

Each of these research streams focuses on FLOSS as a process/mechanism for developing software in distributed environments where the contributors may never meet and drawing on existing legal approaches to intellectual property rights in a novel way. As such, FLOSS differs in important ways from many earlier studies of the process of software development (e.g., Brooks, 1995; Ciborra, 1996; Kidder, 1983).

Our concern in this paper, however, is on properties that FLOSS associates with rather than the process by which they are produced. We therefore focus on the software produced by a particular FLOSS project. We begin by noting that software can exist in a variety of forms, including source code and executable binaries. Software, however, is more than just a thing that happens to be produced, it is something that people engage with and use or not use: it forms a part of their life-world. As such software must be understood as a thing that presents a 'constitutive entanglement' between the material and the social.

The concept of infrastructure forms another important aspect of our study. Many of the most successful FLOSS projects have an infrastructural nature, rather than being stand-alone systems. The inter-connectedness of such systems adds complexity to our understanding of FLOSS because questions of multiplicity apply both to the software itself and the environment within which it exists.

In this paper we draw on an empirical case of a FLOSS project, the Digital Business Ecosystem (DBE). Its aim was to provide an information infrastructure that would address particular societal goals for small and medium enterprises (SMEs) in Europe and elsewhere. The project has drawn inspiration from physical and biological concepts of self-organisation and evolution to produce a technological platform that will facilitate the flexible composition of software services. The evolutionary aspects of the DBE set it apart from similar commercial systems as does the way that the DBE has been explicitly designed around FLOSS principles with all the governance issues this implies.

The DBE engaged people, and therefore came into actual being, by possessing two important characteristics or aspects: It was a potential *idea* of a yet to come — but still absent — 'evolutionary' and open source infrastructure; and it was an iterative and visibly changing open source *project*, existing as a steady stream of releases. That is, the thing itself, the DBE, did not exist as a clear cut 'object', but rather we see it in terms of its productive potentiality and see its iterative changes as something that software objects 'possess' (Latour, 2002).

To understand these aspects of the DBE, we draw on insights on how to understand the software as an 'object' and we draw particularly on the authors Law and Singleton, who provide a useful vocabulary for distinguishing different kinds of change that objects can undergo (Law & Singleton, 2005). For Law and Singleton, fluid objects are those that exhibit incremental change. This change is often gentle, but can occasionally push the 'boundaries of instability' when change can be less gentle and may appear somewhat discontinuous. Fire objects, in contrast, exhibit sudden, discontinuous change that is directly related to the consideration of an *absent other*, a potentiality. We argue that our case study exhibits features of both

fluid and fire objects and this highlights important epistemological (how can we know them?) and managerial (how do we manage them?) challenges for our case study. We suggest that acknowledging these issues helps clarify our understanding of FLOSS software more generally.

We selected the DBE case because it allowed us to expose particular problematics concerning the materiality of software. Software objects are able to retain their identity even though they go through multiple changes. FLOSS in particular is characterised by versioning and frequent releases. The DBE case is also an information infrastructure, and infrastructures also share this ability to retain their identity despite multiple changes. In addition, the evolutionary environment, a key feature of the DBE, was absent for a large part of the engagement activities presented in the paper.

A single case study constitutes a potential limitation of our study, since our analysis seeks to produce general results concerning the nature of software produced by FLOSS. Our case reveals software issues related to materiality, change and infrastructures. We argue that while these insights apply to most forms of software, they are particularly significant in the case of FLOSS given the many ways in which FLOSS differs from more traditional software development and the infrastructural nature of many FLOSS systems.

The structure of the paper is as follows. Section two introduces the research sensitivities about the nature of software and infrastructures that guide our research. This is followed by a section that describes the DBE case and our fieldwork within the DBE project. We then introduce Law and Singleton's vocabulary for describing fluid and fire objects and then use these to understand the engagement with the DBE. We end with a discussion of the implications of our analysis of the DBE case for understanding FLOSS more generally.

Research sensitivities

The nature of software

It would seem reasonable to expect that fields that are particularly concerned with the development, use and management of software based systems might have a strong conceptualisation of the nature of software. For example, the Association for Information Systems states that its mission is "to advance knowledge in the use of information technology to improve organizational performance and individual quality of work life" (AIS, 2007), yet a recent survey of one of the leading journals in information systems (Orlikowski & Iacono, 2001) found the field to be lacking a proper conceptualisation of the IT artefact (including software). Indeed, according to this study, 25% of the articles in the journal *Information Systems Research* had a 'nominal' (i.e., absent) view of the IT artefact. Ayanso et al. (2007) update and broaden this survey and find similar results.

Where Information Systems researchers have attempted to define the nature of software, their definitions typically focus on its abstract nature. For example, Zmud (1980) suggests that software "consists of abstract sets of rules that govern the creation, transfer, and transformation of data. Initially existing solely as an idea, it is iteratively refined, becoming visible at its completion" (p. 45). These views, which focus on the technical expression of code (Ratto, 2005) have been subject to extensive critique and enable us to develop a more nuanced view of what software might be.

A first step towards understanding the software that FLOSS produces is to

explore the 'material' properties of software. At one level, software can be seen as something without material properties, although it can be represented in a variety of formats including as electronic pulses in memory, as textual markings on a printout of source code, as physical marks on, for example, a CD.

Software does, however, have material effects that, as Latour (2003) suggests, "cannot be defined impartially in front of judges without generating fistfights in the courtroom" (p. 37). It is something that people work on, it can be exchanged (at least its representation, for example, as source code). Indeed, FLOSS explicitly uses this ambiguity of materiality/immateriality in its processes. FLOSS licenses, arguably a defining feature of FLOSS (Chengalur-Smith & Sidorova, 2003), freely permit others to make copies of the expression of ideas (i.e., the software) and develop them for their own purposes (GPL, 2006). The digital nature of software, especially if it is available in source code form, as FLOSS requires, means that use by one person does not deprive others of it. Digital goods do not suffer from the traditional tragedy of the commons (Gordon, 1954; Hardin, 1968).

This ambiguity about the materiality of software has led some to suggest that software should perhaps not even be conceptualised as some form of object, instead suggesting that we view software systems as 'configurational technologies' (Fleck, 1999) made up of both technical and non-technical components geared to the needs of individual organizations, as Gestell (Ciborra & Hanseth, 1998), i.e., "the ways through which the ordering and setting up unveils what is extant as standing reserve of resources (including human) made available for future deployment" (p. 320), as narrative networks (Pentland & Feldman, 2007) or even as expectations that can shape future action (Brown & Michael, 2003; Swanson & Ramiller, 1997).

If, however, we remain close to the software in itself, Ratto (2005) suggests that the implicit differentiation between the expressive and functional aspects of software is unhelpful, as software often expresses normative positions about users, programmers and tasks in its 'functional' elements. In so doing he is echoing the views of Orlikowski (2000) who follows Jean Lave's distinction between 'cognition in practice' and 'cognition in the head' to differentiate between the *technological artefact* and *technology–in–practice* (Orlikowski, 2000: 408). The same technological artefact might therefore be used by different users in different ways (including non-use) and this presents a potential solution to the ongoing tension between those who believe that software must have some essential features that structure and constrain action (e.g., Kallinikos, 2002; Klein & Kleinman, 2002) and those who see strong methodological reasons for arguing against such essential features (e.g., Cadili & Whitley, 2005; Grint & Woolgar, 1997).

This emphasis on technology–in–practice highlights the role that software plays in the life-worlds of its users, it is something that people engage with, or not, need to be inspired by, or not. As such it implies a 'mutual entailment' or 'constitutive entanglement' between the material and the social (Barad, 2003: 820; see also Latour, 2007; Orlikowski, 2007: 1437).

Therefore, in addition to the 'features' of software itself, this line of reasoning suggests that we also need to understand software as something that people 'engage with' and, indeed, this notion of engagement forms the basis of the empirical work presented below.

The 'constitutive entanglement' of the software artefact with its use is nicely illustrated in the question of what counts as 'working' software. MacKenzie (1987) presents four responses to claims that determining whether a technology like software is working is unproblematic. First, he argues that many disagreements take place during the design phase, whereas the criterion of working is an ex post facto one; second, even what counts as working is problematic (Collins & Pinch, 1998); third, the range of factors that will typically be required for a technology to work (social, economic and technological) is so large that it may not be obvious what the cause of failure is; and finally that a working technology does not necessarily confirm the rightness of every decision taken in its design (MacKenzie, 1987: 213–214).

The 'intra–actions' between software and its use often remain invisible because of the infrastructural nature of many systems (Mackenzie, 2005: 72) and the next section explores how ideas from information infrastructures can help develop our understanding of FLOSS.

FLOSS and information infrastructures

Information infrastructures are generally understood to consist of standardized systems and data, as well as formal communications mechanisms (Star & Ruhleder, 1996). They are often classified according to their reach and scope in terms of the number of activities they support and the type and variety of activities supported.

Information infrastructures reveal a number of interesting characteristics that are not always apparent from their surface descriptions (Ciborra and associates, 2000). For example, an information infrastructure deals with questions of universal use and access and as such requires high levels of standardization from all potential users of the system. Interoperability between systems is required and this has implications for the flexibility, resilience and security of the system. Infrastructures must also be able to cope with the dual constraints of local variety and centralised planning (Hanseth, Monteiro & Hatling, 1996).

Less straightforward aspects of infrastructures include the fact that they are, effectively, embedded into the systems that use them and this raises important questions of transparency and reach. Infrastructures rapidly become linked to conventions of practice and effectively become a learned part of membership of an organization that uses an infrastructure.

Another key but not immediately apparent feature of infrastructures is that they are always built on an installed base, on the basis of what existed previously. Infrastructures are never built from scratch and they can never be changed all in one go. At a trivial level, switch over is always going to take a finite time and, for most systems, the introduction of a new infrastructure will be phased over a period of months or even years, as new equipment and processes are introduced, with associated periods of retraining and organisational adjustment.

This means that any infrastructure development project will never cover the whole of the infrastructure but rather will need to be developed in conjunction with the constraints arising from existing aspects of the infrastructure. It is, therefore, very difficult to determine in advance what the boundaries of the infrastructure will be. Similarly, it is not straightforward to determine which parts of an infrastructure can be dropped once replacement elements have been introduced. There are many examples of infrastructure code that

contains numerous elements that have been superseded but which remain in place because of the desire not to affect other code that is successfully running.

The relationship between software and FLOSS

At one level, FLOSS is 'just' software. However, FLOSS differs from traditional software development in terms of its entanglement with the user and developer communities which shape the software that is being developed. A high priority in FLOSS communities is being responsive to the needs of the developer base, supporting and facilitating their business use of software components. This is often reflected in processes of strategy formation in FLOSS communities which are frequently modelled explicitly on democratic ideals concerning inclusion and transparency.

Therefore, whilst the FLOSS artefact shares many similarities with software more generally, FLOSS software–in–practice has a number of distinctive features in terms of its development and use because of this entanglement with its user/developer community.

Research question

Implicit in both the review of the nature of software and information infrastructures is the sense that both are open to change whilst also remaining 'stable'. We therefore argue that to understand the software that FLOSS produces we need a vocabulary to address how that software changes (in terms of its immaterial/material aspects, in terms of its use/non-use and in terms of its impact on the infrastructural nature of many systems).

Through our empirical research we seek to move beyond the empirical truism that complex systems have various elements that appear stable/coherent only as a result of much (often hidden) work (e.g., Latour, 1996).

Questions about the nature of 'objects' like software have been widely studied including in science and technology studies, with some interesting work arising in the post-ANT literature. We draw on this literature and the vocabulary it presents to better understand the nature of the changing object that is FLOSS produced software.

We do this by drawing on a large FLOSS project we were involved with, the Digital Business Ecosytem (DBE). Our research seeks to determine the insights about the nature of the FLOSS software object that we can draw from studying the DBE as an infrastructure explicitly based on FLOSS principles.

The Digital Business Ecosystem

Methodology and methods

The digital business ecosystem (DBE) is a concept, a European project and an infrastructural technology (DBE, 2007). The aim of the DBE is to provide a flexible, distributed infrastructure to tie economic development to the region, supporting local trade and industry through the development of software. The project has drawn inspiration from physical and biological concepts of self-organisation and evolution to produce a technological platform that will facilitate the flexible composition of software services. The evolutionary aspects of the DBE set it apart from similar proprietary models such as Microsoft's .Net or SAP's forthcoming business process 'appli-structure', as does the fact that it has been designed as a non-proprietary public infrastructure based around FLOSS principles with all the governance issues this implies (Darking, Whitley & Dini, 2008).

Although the DBE is funded as an European research project, the innovation ecosystems cluster in the EU is equally concerned to ensure that projects like the DBE combine useful scientific advances with major contributions to practice, i.e., a concern to not just deliver technological artefacts. In the case of the DBE, this meant ensuring that SMEs became actively engaged with the DBE. To achieve this, a number of 'regional catalysts' were responsible for co–ordinating the engagement activities and involving local SMEs.

Our involvement in the project was as participant observers, actively involved in, and studying, the process by which small and medium sized enterprises in three regions of Europe (Tampere, Finland; Aragon, Spain; West Midlands, UK) became engaged with the DBE.

The process through which data was collected for this research involved being participant observers at engagement workshops and meetings across the three DBE regions. This fieldwork activity was supported by a programme of interviews with regional catalysts and both actual and potential SME drivers. This last point helped us to understand why some SMEs lost interest as well as the motivations for others to become involved with the project.

The fieldwork reported in this paper associated with the engagement activities was undertaken by the first author, who was a full-time research officer on the project. The research involved attending the DBE engagement events that took place between February and July 2005. In addition, for the engagement study, we carried out 17 semi-structured interviews in each of the three DBE regions (each interview lasting between 1–2 hours). From 'first contact' to formal engagement, the aim was to describe how the interest of driver SMEs in the FLOSS infrastructure was captured and then sustained.

The DBE was a 3-year project and we were engaged as researchers for the final 2 years. Our fieldwork refers to a six-month period during those 2 years in which key aspects of the technology and its use were revealed. Our reason for selecting this 6-month period is that it was during this time that SME engagement in the project was sought and the period was therefore when issues of what it was they were engaging with became important, rather than the early iterations in the development of the artefact itself.

Interviews were recorded and reviewed by both authors independently. The normal project reports and meetings associated with an EU funded project of this scale provided further resources and in addition a number of blogs emerged from participants in the engagement process (e.g., Bergius, 2005; Konda, Shelton & Bayon, 2007) which provided spontaneously generated alternative representations of the events with which to compare the impressions of the researchers.

Once three target groups were identified as part of its engagement strategy (Drivers, Users and Implementers), a significant shift in DBE engagement priorities was agreed by the project. Instead of focusing on recruiting user SMEs (those companies who would use services running on the DBE infrastructure) engagement efforts were focused on driver SMEs (those companies who would provide services) and on influential regional actors such as policy makers. The targets set for regional catalysts were to recruit 3–5 driver SMEs by the end of the first 18 months of the project.

The process of engaging the SMEs with the DBE project raised many and varied issues from a rich and wide ranging set of perspectives and these are de-

tailed elsewhere (e.g., Whitley & Darking, 2006). For this paper, we use this experience to ask what it was that these SMEs were supposed to be engaging with and our focus is on the software object itself and its material practice.

The DBE technology

Although the DBE in its totality encompasses a range of stakeholders, social networks and regional actors, as a technological infrastructure it is comprised of a group of FLOSS projects explicitly linked together by 'light' architectural principles so as to avoid the technological lock-in that infrastructures often face. These principles govern not only the choice of languages, technologies and protocols used, but also the ways in which projects and technologies are integrated.

Broadly speaking, the principles in question stem from a distributed, flexible, open standards approach that seeks to maximise the ability to couple and uncouple component parts. The high degree of abstraction in the overall architecture with its emphasis on 'meta-level' design has been an important element and was a strong selling point for many FLOSS developers and small software houses. By ensuring that all components are essentially dispensable without changing the overall shape of the infrastructure, the way is left clear for new and better solutions to emerge and take their place within the DBE.

This creates more opportunities for developer communities to shape the infrastructure according to their requirements and in line with new developments in the wider software environment. Each and every element of the DBE is created under FLOSS licensing and the core network component has been created as an individual project. When the DBE project first began there were around 40 individual projects worked on by a combination of distributed and co-located teams of developers based in universities and in both large and small technology companies. Gradually these projects consolidated into 3 main areas incorporating these many smaller components: a development environment or service factory (DBE Studio, 2007); a peer–to–peer based execution environment (FADA, 2007; Swallow, 2007); and an optimization facility known as the evolutionary environment (EveNet, 2007). The DBE can be regarded as a distributed middleware that exists above the internet protocol (IP) layer. The DBE run-time environment is a collection of server–clients (ServEnts) and because both end-points are controlled the environment can rely on different transport protocols. Currently, SOAP (Simple Object Access Protocol) is being used but this is simply a starting point and there is nothing to prevent a more efficient binary protocol from being introduced by community developers at a later date.

The DBE's dynamic, multi–layer architecture makes it service-oriented rather than address-oriented, meaning that the service follows the user as the user changes devices. The DBE can be understood as three dynamic and distributed environments or as three sets of local components that allow the individual user to create/describe software services, expose or consume services and issue service search requests. Each instance of the server/client constitutes a node in the dynamic peer–to–peer network through which the infrastructure operates. The repository of services itself is designed to be distributed across local distributed databases (known as 'habitats' in the DBE ecosystem model) to respond to local requests for services. The optimization facility can pick up patterns in service requests and demands

across different 'habitats', automatically distributing services across the infrastructure as and where the need arises.

Every element of the DBE requires a critical mass of users. Network nodes are only created when the infrastructure is used by companies either posting or searching for services. The possibility of reaching new markets by exposing services can only be realized if there are sufficient potential clients searching the infrastructure and the distribution of services across habitats will only be optimized if enough relevant patterns of usage are identified. For this reason, engaging potential users and service providers in the development of infrastructure is key.

Fieldwork

The beginning of fieldwork, which took place in Finland in February 2005, was timed to coincide with the first in a programme of training/engagement events. Whilst there were still no technological components of the DBE to show SMEs, this workshop was designed to focus specifically on the technological concepts and architecture of the DBE. Following this event, the researcher attended every training or recruitment event that took place from this point until mid-June 2005. Further details of the engagement experience are given in Darking and Whitley (2005).

The period in time that this research refers to is a distinctive one because it depicts a period of transition where engagement in the DBE as a technological entity went from being purely conceptual to something tangible. Following actor-network theory, it is not simply status or weight of numbers that governs why a particular preference or point of view is significant (Latour, 2005). It can be the part that point of view plays in achieving a stable network of associations. For example, we found that the first contact any SME had with a DBE technological component was a significant test of credibility and therefore the opinions and feedback offered at that moment were important, even though they concerned just one SME.

According to actor-network theory, likely sites of engagement between users and new technologies are often easy to pinpoint. The work that goes into processes of group formation and enrolment are almost always conspicuous, as it not only involves people but also material and symbolic resources like booking meeting rooms, sending out invitations, ordering refreshments, drawing on personal contacts etc. (Latour, 2005).

In the DBE, the programming of engagement and training events were particularly important. These events marked moments when, for the first time, project machinery, technological components and SMEs were brought together. The social aspect of these networking and dissemination events and the co–learning that inevitably ensued was valuable to all event participants and sometimes featured in their personal blogs. For example, Bergius (2005) writes about the excitement of running the first DBE services over the 'FADA' decentralized peer to peer (P2P) network. Innovative new ideas and associations came from sharing and questioning business, technology and policy ambitions.

Throughout the period of study, however, an important element was missing from the picture: the DBE technology. This meant that interest in the DBE often had to be generated before technological components were 'physically' available for inspection. This is not an uncommon situation when it comes to the dissemination of new technologies as potential users are commonly asked

to engage with a concept, an idea of what a new technology is capable of before they can see it for themselves (Borup, Brown, Konrad & Van Lente, 2006). By definition, it is almost always too late to start engagement activities at the point when technological components are finalised and so pre-emptive engagement action is invariably required. However, the reality of this situation was that early recruitment events lacked *any* applied examples or technological demonstrations. This was a 'pre-prototype stage' where even communicating the basic concept of the DBE presented difficulty due to the advanced nature of the technology.

High level 'scientific' or 'business' overviews were often met with a 'so what?' attitude from SMEs. There was often criticism at engagement events and in SME interviews that project partners were indulging in marketing speak. For example, the idea that the DBE was a 'unique technology' that would 'revolutionise European software development' quickly drew criticism. Presentations that focused on higher order concepts rather than what the technology would 'do', coupled with the fact that the technology in question did not exist in any appreciable, tangible form, led to accusations that the project was attempting to sell 'vapourware'. In order to progress understanding, SMEs would often fall into using metaphors and similes, or else they would seek to cross-reference functional aspects of existing technologies in order to build a picture of what the DBE infrastructure would do. For example, respondents spoke of trying to make sense of the DBE in terms of existing technologies like "P2P protocols and intelligent networking tools" (Bergius, 2005). Mechanical metaphors, particularly those describing the physical working of an engine were often favoured and acted as substitutes for the distinctly intangible infrastructure they were trying to understand. For example, whilst SME participants were content to listen to sessions on architectural design and philosophy up to a point, developers then wanted to "get the bonnet up and see what's under the hood" insisting that in terms of integrating their services with this so far non-existent infrastructure "the devil is in the detail" (Darking & Whitley, 2005).

At one level, it could be argued that many of the difficulties faced by the DBE during the engagement process were similar to those experienced by any innovative project. Interestingly, however, when faced with such problems the SMEs made repeated and forceful requests for 'tangibles' of the FLOSS process — documentation, release dates, components to test, code to compile; they wanted to be able to 'see' something of the technological object. The absence of these elements hindered the process of gaining the trust of SMEs but it did not quash their interest. For some, when asked why they had remained involved in the DBE the answers would mention the architectural principles of the DBE. With its 'meta-approach' to standards, languages and ontologies in the infrastructure as well as the evolutionary aspect of the service evaluation, the potential this architecture suggested for 'levelling the playing field' with respect to small and large software companies was something that carried wide appeal. For others, this was a question of software design methodology. Some partners wanted to see the finished product whilst those familiar with FLOSS methods were happy to proceed with technological components that were works–in–progress although questions were raised about whether the project was sincerely open.

Early engagement events caused those responsible for DBE SME engagement to re-evaluate their strategy. Whilst an 'evolutionary' approach to SME engagement allowed plenty of scope for variation in outcomes and regional strategies, it offered little concrete support for planning what to do and how. First encounters with SMEs suggested that engagement should not be conceived of as a top-down program of research dissemination. Such an approach could not reflect the realities of an engagement process in which development and understanding of the DBE would happen gradually *alongside* the development of DBE technical components which were slowly released.

However, inspecting technical components alone would not yield more understanding. For engagement to be successful, SME developers had to see a clear business use for the DBE. With no prototype to demonstrate and no existing business case to cite SMEs would often begin discussing the possibilities that generating, exposing and combining e–business services using the DBE could open up. Using their own novel ideas for existing or imagined e–business services they would start to brainstorm between themselves, rapidly formulating connections and associations that the DBE could support. A recurrent example came from the SME developers who were frequently commissioned by clients to carry out bespoke integration work, joining together and customising inter-organisational Enterprise Resource Planning systems. By openly discussing individual e-business services in front of participants working in other business sectors, the potential for linking services would strike the SME developers. One observer remarked after one such brainstorming session that he could almost see "light bulbs going on" as individuals suddenly understood what could be possible in terms of developing a market for their services outside the boundaries of their own country or sector (Darking & Whitley, 2005).

Strong feedback loops, an iterative approach to the scheduling and design of future engagement events and the emergence of a fixed group of driver SMEs permitted focus to shift away from initial expectations of finished components and proven business cases to building relationships. Engagement events developed an implicit focus on generating trust and establishing the right conditions for group-based co-learning as opposed to top–down dissemination. Conceptual understanding of how the DBE would alter business and technological practices continued to develop, eventually forming a strong framework for collaboration. As technological components were released engagement events increasingly took the form of workshops where SMEs would experiment with linking their legacy services to the DBE infrastructure via the ServEnt and carrying out 'hello world' tests over the FADA network. Understanding the infrastructure as a dynamic context for innovation opened up possibilities for new associations that were not dependent on geographical proximity. This allowed SMEs to envisage their services in relation to a new paradigm of associations reaching out new potential markets for their services.

A vocabulary for understanding the software object

From the description of the engagement activities it is apparent that there was no single, stable software 'object' that the SMEs were attempting to engage with. In order to make sense of the empirical data we draw on a recent paper by Law and Singleton (2005) (henceforth L&S)

that reports on their difficulties in mapping the 'trajectories' of alcoholic liver disease. Alongside uncertainties about the quality of their research and possible inadequacies in the management of the hospital, L&S highlight two other strategies for addressing the difficulties they were facing.

Their epistemological strategy suggests that some of these problems could have arisen because of differing perspectives on the situation in a manner similar to notions of interpretative flexibility (Pinch & Bijker, 1987) that have been applied in both strong and weak forms to technological objects. Their other strategy is to question the ontological status of the thing they are studying: what exactly is the nature of the object? That is, the object is not simply the same and subject to multiple interpretations, it is in fact different, according to the multiple realities that are enacted into being. It is this strategy that we adopt to understand FLOSS in the case of the DBE.

When they reconsider the nature of the object, their argument passes through a number of stages. They draw on recent, post-ANT studies that develop Latour's original idealised notion of *immutable* mobiles. Immutable mobiles are often seen as mechanisms of long-distance control that are able to maintain their shape despite being part of many (network) relationships.

Recognizing that, in practice, few mobiles are in fact truly immutable, L&S next present the notion of *mutable* mobiles (Moser & Law, 2006) which they illustrate using de Laet and Mol's (2000) description of a water pump in Zimbabwe. The pump changes constantly both in form and function as it is moved between locations and repaired with ready-to-hand materials as parts of it breakdown: "it is something that *both changes and stays the same*" (Law & Singleton, 2005: 338).

The fluidity of the Zimbabwean water pump extends far beyond the ability to replace bolts with steel bars, or the leather seals with a bit of an old tyre (de Laet & Mol, 2000: 238–242), the pump is also fluid in terms of what it produces (exactly how pure must the water be for the pump to 'work' (pp. 242–245)) and how closely the local community must be involved in the creation and maintenance of the pump (pp. 245–247). As such, the pump also shares some of the characteristics of an infrastructure. The fluidity of the pump is therefore enabled by the extensive network that the pump is engaged with (Law & Singleton, 2005: 338).

So far, Law and Singleton's trajectory mirrors that of many authors seeking to make sense of innovations. They recognise that objects often change over time and present a useful metaphor ('fluid') for this process. The fluid metaphor is normally used to describe gentle changes to the object but can include some of the wilder changes that an object undergoes when it reaches the 'boundaries of instability' (Stacey, 2001).

However, the final stage of their analysis provides particular insights that are relevant for making sense of the DBE engagement process. This introduces a post-ANT consideration of issues of invisible work and the colonization of the other (Lee & Brown, 1994). They argue that there is a sense of safety in ANT where networks are built around resilient actors, immutable mobiles or perhaps mutable mobiles that nonetheless change fluidly. The unknown other is often not the focus of these networks which tend to centre themselves around powerful, influential actors. For L&S, as for other critics of ANT (e.g., Star, 1991), this is unsatisfactory, not only from a socio–political point of view but also because the 'not present' can have a huge impact on the shaping of networks. L&S

present the example of a British aircraft company that engineered features of an aircraft wing specifically to cope with flying situations that would occur in an European war against the Russians. 'The war' and 'the Russians' never became a reality but they were still influential in the design of the aircraft. Therefore, "we cannot understand objects unless we also think of them as sets of present dynamics generated in, and generative of, realities that are necessarily absent" (Law & Singleton, 2005: 343). They label such objects as fire objects because "fires are energetic and transformative, and depend on difference — for instance between (absent) fuel or cinders and (present) flame. Fire objects, then, depend upon otherness, and that otherness is generative" (p. 344).

According to the vocabulary presented by Law and Singleton, there is a range of ways in which a current object may point at a future object. The future object may evolve fluidly from the current object. Alternatively, the future object may be discontinuous from the current object specifically because of the consideration of the absent other. In the next section, we argue that FLOSS software as exemplified by the DBE can be more than just a fluid object and that instead the DBE can best be understood in terms of it being both a fire and fluid object.

Understanding the DBE

In order to understand the technological object, L&S argue that we must be prepared to accept "that a fluid, shape-shifting and name-changing object is indeed a conceivable possibility" that is "not ruled out by prior methodological commitments to particular and limited versions of clarity" (Law & Singleton, 2005: 340) yet their paper presents two distinct and distinctive concepts: fire and fluid objects. A common theme in much of the post-ANT literature is a concern not to reify and solidify particular concepts such as strategy (Neyland, 2006) or even actor-network theory itself (Law, 1998). There is therefore a tension: is it possible to not reify fire and fluid objects and yet still be able to highlight and draw on their distinctive characteristics? Although their paper suggests that objects could either be fluid objects *or* fire objects, our study of the process of engaging with the DBE suggest that many FLOSS projects might actually have characteristics of both.

In terms of gently changing shape, many aspects of the DBE as an examplar of FLOSS clearly satisfy the criteria for a fluid object. Perhaps the most straightforward illustration of this can be seen in the FLOSS development process. In common with most FLOSS projects, many parts of the DBE software infrastructure are continuously changing and it is this process of change that will remain central to the sustainability of the infrastructure (Feller & Fitzgerald, 2000).

For example, Table 1 below shows part of the releases log of the DBE Studio (one part of the DBE infrastructure). As the table shows, the software changed (mostly 'gently') approximately every two weeks shifting from being more 'concept than technology' to more 'technology than concept'. Indeed, software engineering has developed a numbering protocol for illustrating whether the changes between versions are minor or major and this versioning is one way in which users can relate to the multiplicity of the software. Essentially minor changes are indicated by adjusting the last numbers (e.g., going from version 0.1.1 to version 0.1.2 is a minor change) whilst more significant changes are indicated by adjusting the first numbers

(e.g., moving to version 0.2.0 or more markedly to version 1.0.0). It is common FLOSS practice for this numbering to be used by the programmers to indicate the difference between (even) stable releases and (odd) unstable releases.

In the case of the DBE, Version 0.1.4 is announced with the message "Big thanks to all that made the *fast turn about* on both reporting and fixing bugs!! :-)". Version 0.1.7 is announced with "This release mainly contains bug fixes and *minor* changes" with a similar announcement for Version 0.1.8, whereas the release of Version 1.0.0 (February 2007) talks of "An *all–in–one* release of the DBE Studio 1.0.0 is available for download from our Sourceforge site. This includes an Eclipse SDK (Windows) distribution with the required GEF, EMF, JEM and WTP feature dependencies" (all extracts taken from DBE Studio, 2007 emphases added). Whilst the release of version 1.0.0 represents a larger jump than the earlier versions, the jump is not a fire–like jump as it does not incorporate the absent other.

If we use these fluid changes as a basis for understanding the DBE and argue that the DBE and perhaps FLOSS more generally are best characterized as fluid objects, then this does not explain the problematic engagement process we saw with the DBE. There is now extensive experience of developing open source projects and if the DBE was *simply* a fluid object, then questions of SME engagement with the DBE would simply have been those that any FLOSS project would encounter and the contribution of science and technology studies to FLOSS could rest with identifying the implications of fluid objects on FLOSS development. Moreover, the empirical evidence from the SME engagement strategies indicates that for many of the SMEs, the FLOSS aspect of the DBE engagement process was relatively unproblematic. Indeed, the calls by a number of developers to make sure that updates to the software were made available regularly and that support requests and bugs were handled promptly (Darking & Whitley, 2005) suggested that for some, the DBE was not fluid enough.

The infrastructural nature of the DBE also does not explain the problems with the engagement process. As an infrastructure, the DBE was explicitly built around changing and staying the same. The design of the system, based around FLOSS licensed components, meant that

Version	Date of Release	Days since last version
Version 0.2.0	2006-02-28 06:54	34
Version 0.1.11	2006-01-25 03:15	16
Version 0.1.10	2006-01-09 15:43	19
Version 0.1.9	2005-12-21 09:10	16
Version 0.1.8	2005-12-05 08:26	14
Version 0.1.7	2005-11-21 09:17	5
Version 0.1.6	2005-11-16 03:54	20
Version 0.1.5	2005-10-27 15:21	2
Version 0.1.4	2005-10-25 15:27	13
Version 0.1.3	2005-10-12 13:14	1
Version 0.1.2	2005-10-11 15:47	0
Version 0.1.1	2005-10-11 15:41	6
Version 0.1.0	2005-10-05 19:35	

Table 1. Version information for DBE studio, taken from (DBE Studio, 2007).

it was always intended that elements of the DBE could be replaced at a later date, without the DBE 'changing' (for example, it would be possible replace the SOAP protocol with a more efficient binary access protocol).

In many cases, it was other aspects of the DBE that affected their engagement with it. The vision for the DBE project for both the EU and project members was much more than simply the development of non–proprietary service infrastructure for SMEs to use (Darking & Whitley, 2005). The engagement strategies could not therefore simply rest upon the provision of a series of smoothly developing software tools that mimicked existing commercially available alternatives. They could not, therefore, put–in–stone too many of the distinctive elements of the DBE that had not yet, at that time, been developed into fully fledged aspects of the ecosystem.

The process of engagement with the DBE also had to incorporate the "realities that are necessarily absent" (Law & Singleton, 2005: 342) as "not everything can be brought to presence" (p. 342) and the DBE is performed by the "enactment of *different objects* in the different sets of relations and contexts of practice" (p. 342).

In the context of engagement, one of the most conspicuous examples of absence was the ecosystem element of the project. Drawing on the work of the science domain of the DBE project, a key element of the project is the ability of the infrastructure to combine and recombine software services available on the DBE flexibly. Whilst many such trial combinations may not necessarily be viable, a distinctive element of the DBE is this ability to make connections between available services to provide new opportunities for user SMEs to interact.

Any engagement activities with SMEs must therefore account for this aspect of the DBE, as this is one of the long term strategic benefits of integrating services with the DBE and one of the key reasons for the initial EU funding of the project and the whole ecosystems technology cluster (Nachira, 2002). This innovative element could not be present in the earliest stages of the DBE engagement process as it both depended on the practical development and implementation of the Evolutionary Environment (EvE) and the population of the DBE infrastructure with sufficient services for this element to begin to make realistic experiments in combining services.

The case has at its core a central dilemma: the process of engagement with the DBE had to deal with a technological object that was undergoing an intensive process of collaborative innovation. In addition, the DBE as a technological object had, at the same time, characteristics of what L&S label as a 'fluid object' which underwent only minor changes over time and their 'fire object' that was defined, in part, by what was not present and was discontinuous from the existing versions of the technology.

The pre-prototype character of the technology is one way in which this complexity can be seen. The tension between absence and near–presence was a tangible reality for those involved in DBE engagement work. However, in trying to organize training and engagement events, the absence of the technology was instinctively countered by participants through brainstorming activities through which they developed their own sense of how the technology could be integrated with specific business ideas and capabilities both now and in the future. In this way, the DBE was drawn into multiple realities, regardless of its 'physical' absence.

Discussion

One of the primary uses that SMEs saw for the DBE environment was the role it could play in facilitating the bespoke systems integration work that many of them were already undertaking. As an infrastructure founded on flexible design principles, the DBE is application, computer and network agnostic which created an environment for constructing data flows. Providing bespoke solutions to facilitate inter–organisational data exchange or, more commonly, intra-organisational data exchange between discrete enterprise resource systems was a mainstay of the SME consultancy work. Being able to adapt systems components as and when their clients requested this, without having to wait for a new release from proprietary technology companies allowed the SMEs remain agile and customer-driven.

This also helps explain why the project attracted the EU funding. The project was initially driven and funded by an EU vision for how SMEs could develop a new environment for competitive collaboration. This public good, however, relied on both the fluid and fire characteristics of the project. The project was funded, in part, because of its fire potential, but had to be developed and implemented using the known fluid practices of FLOSS.

From our study of the engagement with the DBE we note that too much emphasis on the fluid nature of the DBE could create a formative context (Ciborra & Lanzara, 1994) which could stifle the ability of the SMEs to engage with and incorporate many of the advanced, distinctive features of the DBE that would become available when the project ended: "once designed and introduced into the organization, they tend to evolve along paths that are often unexpected and irreversible, subtly changing the ways people design and carry out their work practices, experiment with alternative arrangements and routines, or implement alternative visions and designs" (Ciborra & Lanzara, 1994: 63).

In the same way, too much emphasis on the not–yet–available future capabilities of the DBE would make it unattractive for SMEs to become involved with the DBE, especially the driver SMEs, as they have a particular desire to work with running code and implementable services.

This tension between the fluid and fire aspects of the DBE raises more general concerns about the nature of FLOSS projects. Fluid objects are becoming increasingly understood in practice. Our study of the DBE has emphasized the physical, material nature of many of these fluid changes. The software developers were happy for the DBE to be changing, but they wanted access to source code, design principles, documentation etc. Therefore, fluid aspects of FLOSS emphasise the material nature of the software; they rely on access to the source code, documentation etc. of the project as it unfolds.

The fire nature of the DBE, however, emphasized the immateriality of the project. While some SMEs were intrigued by the functional, operational aspects of the code, as many again were participating in the project because of the immaterial prospects that the discontinuous next version would offer, e.g., the environment of evolutionary service matching. As a result FLOSS projects that have a fire nature will place an emphasis on the immaterial aspects of the system that cannot be addressed by simply providing access to source code etc.

The fluid and fire nature of the project led some users to fear that the project

would end up as yet more vapourware, as it required a careful mixing of both the fluid and fire elements of the system. If the challenges of reconciling the two were successful, however, then the resulting object would both become more stable and yet also remain open to both more stabilized (immutable) and changeable (fire) versions in the future.

This tension between fluid and fire aspects of the DBE can also be seen in terms of the issues associated with the management of information infrastructure more generally. Infrastructures require, on the one hand, the opportunity for expansion and change and on the other hand the diffusion of and investment in the existing infrastructure which leads to a strong conservative influence that opposes change (Monteiro, 1998).

As our study of the DBE confirms, FLOSS differs from proprietary software objects in its constitutive entanglement with the community-based dynamics of collaboration from which components are developed. Remaining responsive to the needs of the developer base, supporting and facilitating their business use of software components is a high priority in FLOSS communities. This gives rise to a more or less unlimited capacity for change and therefore a more fluid software object that directly reflects the needs of contributors. Welcoming voluntary contributions means there is a greater preoccupation with engagement and with maintaining a synergy between the software object and its developers.

This means that in contrast to software more generally, the FLOSS object changes in an open (fluid) manner with developers taking time to articulate their learning for the benefit of the community, whereas for proprietary software development learning and innovation are often kept 'secret'. The greater role that co–operation and negotiation play in strategy formation for FLOSS communities has implications for the way in which discontinuous (fire) changes are experienced by the developer base. Allowing the software to be altered by its developer base in response to user and business needs opens up potential for innovation; potential that may not be visible from traditional managerial and strategic viewpoints. An approach to governance or strategy that is too rigid and that does not take into account the sensitivities we describe in this paper could therefore have potentially limiting effects on the kind of boundary–crossing innovations that FLOSS technologies are capable of achieving.

For the science and technology studies literature, the DBE provides an empirical opportunity for studying the existence of objects that offer both the characteristics of fluid and fire. This allows us to develop the analysis beyond that presented by Law and Singleton and take their approach further and in particular to explore the managerial and epistemological concerns that such objects raise.

However, more research is required to understand how fire objects are created in practice. In their illustration of fire objects, L&S state that consideration of 'the Russians' shaped the design of the military aircraft wing as it needed more lift than a civil aircraft wing would need (p. 343), but they give limited guidance as to how this consideration of the absent other takes place or how it can be studied, other than suggesting that it is probably best studied qualitatively rather than quantitatively as they do with their emphasis on interviews and case studies. For example, if the absent other is "the elephant in the room", i.e., something that everyone knows about

but does not mention, how can the associations (Latour, 1986) between it and the fire object be demonstrated? If the absent other is widely understood, it may never be discussed in the developer mailing lists and IRC chats, the most common location for data collection in FLOSS projects (Kuk, 2006).

Acknowledgements

The authors would like to thank the special issue and journal editors for their support and encouragement and would particularly like to thank the two anonymous reviewers for their helpful and insightful comments. Special thanks also to all the participants at the special issue workshop in Helsinki for their comments on the earlier versions of the paper. Aaron Martin, as ever, did an excellent job in proofreading the text for us. A part of this work was funded by the Digital Business Ecosystem–DBE FP6 Integrated Project (DG–INFSO), Contract number 507953.

References

AIS (2007) 'About the Association for Information Systems', URL: http://home.aisnet.org/ Last visited 17 December 2007

Ayanso, Anteneh, Lertwachara, Kaveepan & Francine Vachon (2007) 'Diversity or identity crisis? An examination of leading IS journals', Communications of the AIS 20: 660-80.

Barad, Karen (2003) 'Posthumanist performativity: Toward an understanding of how matter comes to matter', Signs 28(3): 801-32.

Benkler, Yochai (2002) 'Coase's Penguin, or Linux and the nature of the firm', Yale Law Journal 112(Winter 2002-2003): 369-446.

Bergius, Henri (2005) 'Motorcycle adventures and free software: A blog', URL: http://bergie.iki.fi/blog/archive/month/2005/6.html Last visited 17 December 2007

Bergius, Henri (2005) 'First look at Digital Business Ecosystem', URL: http://bergie.iki.fi/blog/first-look-at-digital-business-ecosystem.html Last visited 17 December 2007

Borup, Mads, Brown, Nik, Konrad, Kornelia & Harro Van Lente (2006) 'The sociology of expectations in science and technology', Technology analysis and strategic management 18(3/4): 285-98.

Brooks, Frederick P. (1995) The Mythical Man-Month: Essays on Software Engineering (Cambridge, MA: Addison-Wesley).

Brown, Nik & Mike Michael (2003) 'A sociology of expectations: Retrospecting prospects and prospecting retrospects', Technology analysis and strategic management 15(1): 3-18.

Cadili, Sarah & Edgar A. Whitley (2005) 'On the interpretative flexibility of hosted ERP systems', Journal of Strategic Information Systems 14(2): 167-95.

Chengalur-Smith, Shobha & Anna Sidorova (2003) 'Survival of open-source projects: A population ecology perspective', in S. T. March, A. P. Massey & J. I. DeGross (eds) International Conference on Information Systems (Seattle: 782-87).

Ciborra, Claudio U. (ed) (1996) Groupware & Teamwork: Invisible aid or technical hindrance (Chichester: Wiley).

Ciborra, Claudio U. and associates (2000) From Control to Drift: The dynamics of corporate information infrastructures (Oxford: Oxford University Press).

Ciborra, Claudio U. & Ole Hanseth (1998) 'From tool to *Gestell*: Agendas for managing the information infra-

structure', Information technology and people 11(4): 305-27.
Ciborra, Claudio U. & Giovan Francesco Lanzara (1994) 'Formative contexts and information technology: Understanding the dynamics of innovation in organizations', Accounting, management and information technologies 4(2): 61-86.
Collins, Harry & Trevor Pinch (1998) The Golem at Large: What you should know about technology (Cambridge: Cambridge University Press).
Darking, Mary & Edgar A. Whitley (2005) 'Project report 27.2 - Studying SME Engagement Practices',URL: http://personal.lse.ac.uk/whitley/allpubs/DBE2005.pdf Last visited 17 December 2007
Darking, Mary, Whitley, Edgar A. & Paolo Dini (2008) 'Governing diversity in the digital ecosystem', Communications of the ACM Forthcoming.
DBE (2007) 'DBE project website', URL: http://www.digital-ecosystem.org/ Last visited 17 December 2007
DBE Studio (2007) 'The DBE Studio is an Integrated Development Environment (IDE) for the Digital Business Ecosystem (DBE). It includes eclipse plugins that allow business services to be analysed, and corresponding software services to be defined, developed and deployed'. URL: http://sourceforge.net/projects/dbestudio Last visited 17 December 2007
de Laet, Marianne & Annemarie Mol (2000) 'The Zimbabwe bush pump: Mechanics of a fluid technology', Social studies of science 30(2): 225-63.
EveNet (2007) 'Provides a P2P network for guiding the evolution of software services and service compositions over time', URL: http://sourceforge.net/projects/evenet Last visited 17 December 2007
FADA (2007) 'FADA is an autonomous distributed directory of JAVA proxies following the same philosophy than Jini Networking Technology', URL: http://sourceforge.net/projects/fada Last visited 17 December 2007
Feller, Joseph & Brian Fitzgerald (2000) 'A framework analysis of the open source software development paradigm', in S. Ang, H. Krcmar, W. J. Orlikowski, P. Weill & J. I. DeGross (eds) International Conference on Information Systems (Brisbane, Australia: 58-69).
Fleck, James (1999) 'Learning by trying: the implementation of configurational technology', in D. Mackenzie & J. Wajcman (eds), The social shaping of technology (Buckingham: Open University Press): 244-57.
Gordon, H. Scott (1954) 'The economic theory of a common property resource: The fishery', Journal of Political Economy 62(2): 124-42.
GPL (2006) 'GNU Public Licence', URL: http://www.gnu.org/copyleft/gpl.html Last visited 17 December 2007
Grint, Keith & Steve Woolgar (1997) The Machine at Work: Technology, work and organization (Cambridge: Polity Press).
Hanseth, Ole, Monteiro, Eric & Morten Hatling (1996) 'Developing information infrastructure: The tension between standardization and flexibility', Science, Technology, & Human Values 21(4): 407-26.
Hardin, Garrett (1968) 'The tragedy of the commons', Science 162: 1243-48.
Himanen, Pekka (2001) The Hacker Ethic and the Spirit of the Information Age (London: Random House).
Kallinikos, Jannis (2002) 'Reopening the black box of technology artefacts and human agency', in L. Applegate, R. D. Galliers & J. I. DeGross (eds) International Conference on Information Systems (Barcelona: 287-94.

Kidder, Tracy (1983) The Soul of a New Machine (Boston: Little, Brown).

Klein, Hans K. & Daniel Lee Kleinman (2002) 'The social construction of technology: Structural considerations', Science, technology and human values 27(1): 28-52.

Konda, Nagaraj, Shelton, Rod & Victor Bayon (2007) 'Open SOA - Digitial business ecosystem: Blog for the DBE project', URL: http://opensoa.blogspot.com/ Last visited 17 December 2007

Kuk, George (2006) 'Strategic interaction and knowledge sharing in the KDE developer mailing list', Management Science 52(7): 1031-42.

Latour, B. (2007) 'Can we get our materialism back, please?' Isis 98(1): 138-42.

Latour, Bruno (1986) 'Powers of association', in J. Law (ed) Power, action and belief: A new sociology of knowledge? (London: Routledge & Kegan Paul): 264-80.

Latour, Bruno (1996) Aramis, or the Love of Technology (Cambridge, MA: Harvard University Press).

Latour, Bruno (2002) 'Gabriel Tarde and the end of the social', in P. Joyce (ed) The Social in Question. New Bearings in History and the Social Sciences (London: Routledge): 117-32.

Latour, Bruno (2003) 'Is *re*-modernization occurring-and if so, how to prove it', Theory, culture and society 20(2): 35-48.

Latour, Bruno (2005) Reassembling the Social: An introduction to Actor-Network-Theory (Oxford: Oxford University Press).

Law, John (1998) 'After ANT: complexity, naming and topology', in J. Law & J. Hassard (eds) Actor network and after (Oxford: Blackwell): 1-14.

Law, John & Vicky Singleton (2005) 'Object lessons', Organization 12(3): 331-55.

Lee, Nick & Steve Brown (1994) 'Otherness and the Actor Network: The undiscovered continent', American Behavioural Scientist 37(6): 772-90.

Mackenzie, Adrian (2005) 'The performativity of code: Software and cultures of circulation', Information, Communication and Society 22(1): 71-92.

MacKenzie, Donald (1987) 'Missile accuracy: A case study in the social processes of technological change', in W. E. Bijker, T. P. Hughes & T. J. Pinch (eds) The Social Construction of Technological Systems: New directions in the sociology and history of technology (Cambridge, MA: The MIT Press): 195-222.

Management Science (2006) 'Special issue on Open Source', Management Science 52(7).

Mockus, Audris, Fielding, Roy T. & James D. Herbsleb (2002) 'Two case studies of open source software development: Apache and Mozilla', ACM Transactions on software engineering and methodology 11(3): 309-46.

Monteiro, Eric (1998) 'Scaling information infrastructure: The case of next-generation IP in the internet', The information society 14(3): 229-45.

Moser, Ingunn & John Law (2006) 'Fluids or flows? Information and qualculation in medical practice', IT and People 19(1): 55-73.

Nachira, Francesco (2002) 'Towards a Network of Digital Business Ecosystems', URL: http://europa.eu.int/information_society/topics/ebusiness/godigital/sme_research/doc/dbe_discussionpaper.pdf Last visited 17 December 2007

Neyland, Daniel (2006) 'Dismissed content and discontent: An analysis of the strategic aspects of actor-network theory', Science, technology and human values 31(1): 29-51.

Orlikowski, Wanda J. (2007) 'Sociomaterial Practices: Exploring Technology at Work', Organization Studies 28(9): 1435-48.

Orlikowski, Wanda J. (2000) 'Using technology and constituting Structures: A practice lens for studying technology in organizations', Organizational Science 11(4): 404-28.

Orlikowski, Wanda J. & Suzanne C. Iacono (2001) 'Research commentary: Desperately seeking the "IT" in IT research: A call to theorizing the IT artifact', Information Systems Research 12(2): 121-34.

Pentland, Brian T & Martha Feldman (2007) 'Narrative networks: Patterns of technology and organization', Organization Science 18(5): 781-95.

Pinch, Trevor J & Wiebe E. Bijker (1987) 'The social construction of facts and artifacts: Or how the sociology of science and the sociology of technology might benefit each other', in W. E. Bijker, T. P. Hughes & T. J. Pinch (eds) The social construction of technological systems: New directions in the sociology and history of technology (Cambridge, MA: The MIT Press): 17-50.

Ratto, Matt (2005) 'Embedded technical expression: Code and the leveraging of functionality', The information society 21(3): 205-13.

Stacey, Ralph D. (2001) 'The science of complexity: an alternative approach to strategic change processes', Strategic Change Management Journal 16(6): 477-95.

Stamelos, Ioannis, Angelis, Lefteris, Oikonomou, Apostolos & Georgios L. Bleris. (2002) 'Code quality analysis in open source software development', Information Systems Journal 12(1): 43-60.

Star, Susan Leigh (1991) 'Power, technologies and the phenomenology of standards: On being allergic to onions', in J. Law (ed), A Sociology of Monsters (Oxford: Basil Blackwell): 27-57.

Star, Susan Leigh & Karen Ruhleder (1996) 'Steps toward an ecology of infrastructure: Design and access for large information space', Information Systems Research 7(1): 111-34.

Swallow (2007) 'A peer2peer application container that isolates the programer form the peer2peer coding complexity', URL: http://sourceforge.net/projects/swallow Last visited 17 December 2007

Swanson, E. Burton & Neil Ramiller (1997) 'The organizing vision in information systems innovation', Organization Science 8(5): 458-74.

von Krogh, Georg & Sebastian Spaeth (2007) 'The open source software phenomenon: Characteristics that promote research', Journal of Strategic Information Systems 16(3): 236-53.

Whitley, Edgar A. & Mary Darking (2006) 'Object lessons and invisible technologies', Journal of information technology 21(3): 176-84.

Zmud, Robert W. (1980) 'Managing large software development efforts', MIS Quarterly 4(2): 45-55.

Mary L. Darking
School of Applied Social Science, University of Brighton, UK and Department of Media and Communications, LSE
m.l.darking@brighton.ac.uk

Edgar A. Whitley
Information Systems and Innovation Group, Department of Management, London School of Economics and Political Science, UK
e.a.whitley@lse.ac.uk