Social Studies of Science

Global Software and its Provenance: Generification Work in the Production of Organizational Software Packages

Neil Pollock, Robin Williams and Luciana D'Adderio

The online version of this article can be found at:

Published by:
$SAGE
http://www.sagepublications.com

Additional services and information for *Social Studies of Science* can be found at:

Email Alerts: http://sss.sagepub.com/cgi/alerts

Subscriptions: http://sss.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.co.uk/journalsPermissions.nav

Citations http://sss.sagepub.com/cgi/content/refs/37/2/254

S|S|S

ABSTRACT   This paper addresses the seemingly implausible project of establishing a 'generic' organizational information system. This is an apparent contradiction: on the one hand, we are told of the diversity of specific organizational contexts and on the other, we often find *the same* standardized software solutions being applied across those settings. How do generic software packages work in so many different contexts? Science and Technology Studies provides contrasting accounts of how this contradiction is resolved: either stressing the unwanted organizational change that standardized systems may bring; or, alternatively, insisting these technologies can only be made to work through processes of 'localization'. We argue that the focus on specificity versus localization of application contexts draws attention away from enquiring into the origins and characteristics of generic solutions. Through comparing the design and evolution of two software packages we shift the debate from understanding how technologies are made to work within particular settings to how they are built to work *across* a diverse range of organizational contexts. Our question is 'How do software packages achieve the mobility that allows them to bridge the heterogeneity within organizations *and* between organizations in different sectors and cultures?' We describe a set of revealed strategies through which suppliers produce software that embodies characteristics common across many users; what we term *generification work*. One aspect of this process of generification is the configuring of users within 'managed communities', but it also includes 'smoothing' the contents of the package and, at times, reverting to 'social authority'. Our argument is that generic systems *do* exist but that they are brought into being through an intricately managed process, involving the broader extension of a particularized software application and, at the same time, the management of the user community attached to that solution.

Keywords   generification, localization, particularity, sameness, software packages

Global Software and its Provenance:

Generification Work in the Production of Organizational Software Packages

Neil Pollock, Robin Williams and Luciana D'Adderio

Complex organizational information systems do not travel. Berg (1997) suggests that the difficulties in transporting such systems from one place to another arise because they become fixed in 'time' and 'space'. His argument is that software becomes so thoroughly imbued with the local idiosyncrasies of its place(s) of production that it only works at the site(s) for which it was designed and built. Scholars in Science and Technology

Studies (STS) and other fields have spent much time describing how building anything other than the simplest artefact produces this kind of *particularization*. There are dozens of such examples in STS of how manufacturing planning systems, finance sector administrative systems, hospital information systems, and, to use Berg's example, expert systems resist transfer to other settings.[1]

Yet there is a curious contradiction. Despite familiar-sounding stories of failed or problematic technology transfer, there are, of course, many types of software that do appear to be highly mobile. For instance, Enterprise Resource Planning (ERP) systems, the name given to one of the most popular types of integrated organizational information system, are used in diverse places and appear oblivious to the form, function, culture or even geography of organizations.[2] Such has been their ability to transcend their place of production that they are now described as 'generic' or even 'global' solutions.

How are we to understand travelling software through the lens of STS? Some have sought to question their existence, disputing whether there is such a thing as a generic system. According to this argument, a truly global system is a modernist dream: there are no 'genuine universals' in large-scale information technologies (Star & Ruhleder, 1996: 112); and their creation is akin to 'hunting for treasure at the end of a rainbow' (Hanseth & Braa, 2001: 261). An alternative STS approach has been to highlight the effort of local actors in making these systems work in specific local settings (McLaughlin et al., 1999). From this perspective, we might focus on the all too apparent gulf between the software presumptions and actual working practices at the settings where the solution is adopted (as well as the active processes whereby humans repair these deficiencies). Despite these objections from within STS, the notion of a generic technology continues to be a powerful and attractive idea. There are many software suppliers, for instance, who act as if it were possible to build such an object. It is not our intention to refute the rhetorics of technology suppliers who claim to create universal solutions to organizational activities; instead, we intend to take seriously their ambitions and strategies to create such solutions. Rather than focus on the effort of 'localization', and thus highlight the already well researched 'collision' of system and setting, we seek to examine the much less investigated and poorly understood process through which systems are designed to work across many contexts. Indeed, we think it odd that STS has little to say about generic software, given that, as we discuss below, from its earliest days it has concerned itself with how knowledge is made to transcend its place of production.

Why might this be so? Perhaps this relates to a more fundamental problem where contemporary social scientific analyses are not good at thinking about movement (Cooper, 1998). STS is interested in how technologies are translated for new contexts: and, of course, a kind of movement is examined in these studies, but the primary interest is in the process of translation as a matter of *localization*: of how software is both made to work *within* a specific setting and how it transforms that setting. There are

limitations with movement as 'simple location' applied to generic software, to use Alfred North Whitehead's term (Whitehead, 1967; cited in Cooper, 1998: 108). For instance, there is little concern for any transformations in the thing that is moved, such as with the ways in which the software package is explicitly designed (and redesigned) to work *across* settings. We find it odd that there is such a wide-ranging set of terms in STS to describe the way standardized technologies are 'imported' ('domesticated', 'appropriated' or 'worked-around') into user settings, while there is a comparative lack of emphasis on the reverse process through which an artefact is 'exported' from the setting(s) in which it was produced. This is striking since the bulk of organizational software in use today is produced in this way – the same systems are recycled from one context to another. By attempting to develop the beginnings of a vocabulary to capture this exporting, we describe the practice of making software generic (generification work), including its various explicit and revealed generification strategies, as the process of *generification*.[3] Through discussing a number of generification strategies we hope to offer novel or fresh insight into the design and use of software packages.

The design of generic packages differs from earlier software development traditions. Suppliers traditionally developed close ties with customers, the conventional wisdom being that increased knowledge of users would lead to better design. In contrast, generic solution suppliers are said to actively keep users at a distance, fearing that their software will become identified with and tied to specific user organizations and thus not widely marketable (Bansler & Havn, 1996; Williams et al., 2005). Consequently, in the information systems literature, software package construction is conceived of as *design for markets* (Salzman & Rosenthal, 1994; Sawyer, 2000, 2001). Accordingly, it is said that programmers work without concrete notions of users in mind, a process Suchman (1994) describes as being akin to 'design from nowhere'.[4] However, we are sceptical that complex organizational systems can be designed for abstract markets in an asocial manner. To explore this, we present material on the design and evolution of two software packages, and describe how suppliers actively manage users through configuring them within 'communities'. In these groups suppliers control which functionality and whose particularity will be accommodated through various forms of generification work. Before turning to the empirical material, we review how the literature on information systems has dealt with generic systems, and then we turn to relevant work within STS.

Narrative Biases in STS: Localization

The nature of software development has changed in the last 30 years (Friedman & Cornford, 1989). Whereas user organizations once built or commissioned their own software, they now prefer to buy 'commodified solutions'. Initially these were 'low level' software systems (such as operating systems, utilities and application tools), but increasingly they are also

the 'higher level' organizational information systems (such as payroll, procurement and human resources) and industry-specific systems such as those we are discussing (Brady et al., 1992; Quintas, 1994; Pollock et al., 2003). From the point of view of scholars sensitive to organizational diversity, this move is highly implausible, since software packages such as ERP encompass a wide range of organizational activities which, because of their intricacy, are likely to vary from one organization to another (Fincham et al., 1994: 283). In contrast, and buoyed up by the seeming success of these systems, proponents argue that they can be adapted to work in most organizations within the same class and, in principle, across different classes of organizations. In explicating these arguments, scholars point to the similarities that exist between organizations, as well as to the 'flexibility' of generic systems that allows them to be custom-fitted to even the most idiosyncratic of settings (Davenport, 2000). As a rejoinder to these 'universalistic' presumptions, a large body of fine-grained empirical research has pointed to the difficulties adopters have with implementing them, as well as the large levels of unwanted organizational change they require – standardized systems may thus bring risks and unanticipated costs. The aim of much of this research has been to demonstrate that getting these systems to work is an 'accomplishment'; an active process whereby users reconcile the gulf between system and actual work practices (McLaughlin et al., 1999).[5] If they can transfer between settings it is only as a result of this major localized effort; they work because they have been re-designed around the cultures and practices of user organizations.[6]

In our view, the STS literature tends to over-emphasize the collision between specific organizational practices and generic system presumptions at the point of implementation within specific user organizations (see, for example, Walsham, 2001; Avgerou, 2002). This, we would argue, reflects the various narrative biases within current STS and sociology: that contexts of use are always individually different, unique and typified by highly idiosyncratic practices; whereas technologies are 'singular' and 'monolithic'; and localization is the means by which the standard and the unique are somehow brought together.[7] A further concern is that localization studies do not adequately address the longer-term co-evolution of artefacts and their social settings of use. This is not to say that we should view generic solutions as embodying features that can and should be applied in all contexts. We must also resist universalistic accounts and develop a language and set of concepts to describe how generic solutions are designed to pass over organizational, sectoral and national boundaries while embracing aspects of the specific features within these settings. In this regard, we argue that the notion of localization, together with the concept of generification, can be taken further to explain this circulation. Our argument is not that the organizations in which the software circulates are the same; rather, it is that, through various generification strategies, these local sites *can be treated as the same*. How, then, are we to account for those times when the generic systems do actually travel across many contexts (Rolland & Monteiro, 2002)?

From Importing to Exporting

Ophir & Shapin (1991) asked a similar question some years ago in relation to scientific knowledge. This was a reaction to the 'localist turn' in the Sociology of Scientific Knowledge (SSK): scholars, sceptical of the claim that knowledge diffuses because it is 'true', sought to show how the universality of science was both an 'acquired quality' and 'local affair'. They did this by emphasizing how facts were produced with reference to specific places and times, that they were the product of particular communities and that there were tacit practices involved in their production (Knorr Cetina, 1981; Turnbull, 2000; Hanseth & Braa, 2001). Ophir & Shapin's (1991: 15) question was 'If knowledge is such a "local product", then how does it manage to travel with such "unique efficiency"?' Others voiced similar questions at the time and this led to a growth in 'laboratory ethnographies' and an interest in demonstrating just how knowledge *escaped its locality*: this was the claim that knowledge only became universal after contextual features of locality or 'particularity' were *deleted*. Moreover, to 'solve' this problem of how knowledge moved from one laboratory to another, Latour (1987, 1999) introduced various terms such as 'immutable mobile' and, more recently, 'circulating reference'.

While these terms have become commonplace within the STS vocabulary, they also have been criticized. First, much of the criticism objects to the overly imperialistic language used by Latour and other proponents of actor-network theory: 'immutability' seems to suggest that devices remain standardized at the centres at which they are produced, the locales at which they are used, and as they pass through the channels between these places. In particular, the notion of immutable mobile directs attention away from the localized work of adapting an inscription or innovation to a local context of use and setting up the conditions for its effective 'travel' (Knorr Cetina & Amann, 1990).[8] Second, the terms are also criticized for implying that marks of locality are simply deleted. On the first point, and writing some years earlier, Ravetz (1972) had attempted to give a more sensitive treatment of the spread of knowledge by arguing, not for the immutability of scientific knowledge, but for its 'malleability'. Knowledge, tools and instruments, he argued, were widely adopted through processes of 'smoothing'. That is, scientists importing methods or techniques from outside their normal domain would ignore any obscurities or unresolved conceptual difficulties surrounding that object.[9] In terms of the second point, Turnbull sought to build on Latour's work by showing how the local, rather than simply being erased, was often 'aggregated'. He illustrates this through a discussion of the way in which indigenous knowledges spread through a process of bridging:

> I argue that the common element in all knowledge systems is their localness, and their differences lie in the way that local knowledge is assembled through social strategies and technical devices for establishing equivalences and connections between otherwise heterogeneous and incompatible components. (Turnbull, 2000: 13)

In other words, local knowledge diffuses through the creation of 'similarities' and 'equivalences' between diverse sites. Such equivalence-making requires a number of different devices and strategies, such as 'standardization' and 'collective working', some of which we will explore further with empirical material.[10]

The Studies

We analyse two software packages which are at different stages in their 'biography' and characterized by different levels of product maturity and standardization.[11] The first is a student administration system – the Campus Management module (CM) – developed by the German software house, SAP, to integrate with its already highly successful ERP R/3 system. To develop CM, the Supplier had involved a number of universities as the 'surrogates' on which the software would be modelled before it would finally be launched to the wider market as a 'global university solution'. While SAP was new to the higher education sector, it had developed software for unfamiliar settings many times before. The second study is of the student accommodation system PAMS, which was built by a company we call 'Educational Systems'. PAMS was initially designed around the needs of one Scottish University but is now being used by more than 40 other institutions in the UK, and the Supplier is currently investigating the potential market overseas. PAMS has associated with it a growing and active 'user group' that meets regularly to learn about new product developments and petition for the building of further functionality. Whereas SAP already had in place established design methods and processes for software package design, Educational Systems did not; the latter company was new to both higher education and to the development of software packages.[12]

Birth of a Package

The 'birth' stages of the biography of a software package are the most dramatic. In this phase there are few users in place and the large community upon which the package will depend for its circulation is yet to be enrolled. Seemingly, there are many choices influencing the extent to which the package will become 'generic' and therefore attractive to the widest possible groups of users. Suppliers will spend time deciding which organizational practices will be catered for and which will not. In truth, however, and despite the seeming importance of this stage, the suppliers appeared initially to follow a strategy of simply and rapidly 'accumulating functionality'.

Accumulative Functionality[13]

Software packages are designed around a basic organizational functionality, what is sometimes described as the 'generic kernel'. The idea is to paint the organizational reality of adopters onto this kernel by developing numerous

'templates', which users can then choose between and tailor to meet their local conditions. These templates form the 'outer layer' of the package, and are built up over time through interactions with past customers. Suppliers only reap benefits from developing new templates when they are able to use them again and again (thus recouping development costs). In the birth stages, both suppliers found that, rather than simply re-using templates, they were repeatedly forced to modify or build new ones. For instance, Educational Systems found that with each new customer for PAMS, the templates required modification. The Sales Director describes this in relation to the 'Payment Schedule' process:

> When we first wrote PAMS for [Scotia University] they produced a Payment Schedule that gave the student the choice of paying in 3 equal installments (1 per term) or equal monthly installments. The logic was therefore simple in that PAMS added up all of the charges and divided by the number of installments.

However, when they made the next sale to 'Highbrow' university there were some differences which required changes to the software:

> The next customer, [Highbrow], also offered the choice of paying in termly installments, but they massaged the amounts to take 40% in term 1, 40% in term 2, and 20% in term 3, as they wanted to get as much paid as possible before the student ran out of money. We therefore added a tick box on the payment plan to say 'use ratios', and this then gave access to an extra column that allows them to enter the % against each installment.

He describes how they could accommodate the next user with the changes conducted for Highbrow: 'The next customer [Seaside] also produced a termly plan, but used the number of days in each term to compute the amount. Fortunately, the work we had done for [Highbrow] was capable of managing this, as the days in each term could be entered as numbers as well as percentages.' But, once again, when another user adopted the package they were forced to make changes: '[Central] came along. And they offered students a discount if they paid by a certain date, so we had to add another (optional) column that stored the settlement date for each instalment and we added the code to compute the value of this discount.' The Sales Director goes on to describe the modifications required by two further universities: '[City], on the other hand, charges a penalty for late payments. So we added a process that calculated a charge for late payment'; and on the other hand, '[Rural] wanted this banded as their fees change according to the amount owed, so we added extra functions to band the charge according to the value.'

What is clear is that as each new site adopts the package, new and different requirements need to be catered for. Importantly, this occurs not simply in the Payment Schedule process but in all the other templates stored in the system library. The Supplier appeared to be building into the system whatever functionality was asked for. However, it was becoming obvious to

Educational Systems that accumulating and not re-using functionality was *particularizing* PAMS. In the case of the Payment Schedule, for instance, every time a change was made to the template this would be accompanied by a modification to the graphical user interface. A user was then forced to view a screen that included buttons and menus specifically intended for other institutions. As a result, there was now a need for increased training where users were told which options and buttons related to them and which did not. However, this mode of redressing the particularization of PAMS became problematic once the system was made available for operation by students over the Internet. One of the managers describes the problem:

> … how do you get rid of the things that a particular site doesn't want? For example, in our payment process we handle things like 'settlement discount'. Somewhere like [Welsh University] do not use settlement discount but they just ignore the fields on the screen. If you put that on the Web, all you do is end up with calls from customers, from students asking 'Why haven't I got any settlement discount?' When actually the answer is that 'We do not use it, so we do not want to display it'. So how do we get over that?

During the birth stage, then, suppliers are presented with choices. If they continue with the strategy of accumulative functionality, PAMS will become increasingly baroque, locked in to the particular requirements of their specific array of existing users. This realization led to a switch in strategy. As the Managing Director of Educational Systems puts it: 'We are not going to accommodate as much diversity as we have in the past because it constrains our ability to grow and resell'. Any changes we make to the package from now on, he says, will have to have wider applicability: 'When we built change into the software we have always tried to build it in a way that isn't customer specific and we try to always broaden it a bit so that we have functionality that has a potentially wider audience'. During one particular conversation he described how they now try to 'discourage too much diversity'. Yet this presents the Supplier with an interesting problem: how do they continue to make the software attractive to, and, indeed, encourage, a wider range of new users without having to include every demand for new functionality? Importantly, how do they 'discourage too much diversity' without discouraging the users attached to this diversity?

Management by Community

If the software is truly designed to travel, then it seems that the suppliers must avoid dealing with individual users. Indeed, the translation from a particular to a generic technology corresponds to a shift from a few isolated users to a larger extended 'community' (Cambrosio & Keating, 1995; de Laet & Mol, 2000). Moreover, it is through establishing and engaging with the users primarily through such a forum that suppliers are able to *shape* these communities and to extend the process of generification. In other words, through participating in community environments, such as the user-group meetings

and requirement prototyping sessions, individual organizations were often dislodged from attachments to particular needs.[14]

Community Management Strategies

The suppliers had close ties with individual user organizations in the earlier phases, but they felt forced to shift to an alternative form of relationship as the technology matured and the user base grew. The openness of the software that was stressed during initial interactions was reversed: where they had previously negotiated on a one-to-one basis with users, they now appeared increasingly reluctant to differentiate users. Individual conversations about design issues were *shifted* to a more public forum. This shifting out is also demonstrated in the case of SAP, which had elaborate routines for managing its communities (and though the same strategies were visible within Educational Systems, they appeared much less developed). SAP had developed CM by gathering requirements during site visits and from other direct correspondence with users. The problem in accumulating functionality in this way was that they were 'flooded with particular requests'.[15] How might they construct something more generic from these requests? Moreover, if they were to 'discourage diversity', how would users react if they felt their needs were not being met (and perhaps those of a neighbour were)? Thus, there was potential for this problem to become a focus of conflict (and the precious pilot sites on which the future of the product depended might be discouraged or, worse, lost).

Witnessing

During the requirements prototyping sessions, a wide number of potential users were invited to the SAP University in Waldorf, Germany. The reported functions of these meetings, which would last as long as 2 weeks, were to receive feedback on beta versions of the software and to continue the requirements gathering process. It was the latter process that was the most striking. Participants from more than a dozen universities and as many countries were seated in a room. Each appeared determined to spell out in magnificent detail just how *their* particular requirements differed from the prototype on the screen in front of them, or, just as likely, from the view being articulated by their neighbour at the next desk. In the excerpt below, they discuss the storing of student transcripts and whether universities need to store details on both passed and failed courses. A consultant standing at the front attempts to make sense of the comments by scribbling them onto overhead projector (OHP) slides:

> SAP Consultant: Does everyone want the ability to store two records?
>
> America South Uni: We would maintain only one record …
>
> SAP Consultant: Is there a need to go back into history? If transcript received and courses are missing do you need to store this?
>
> America North Uni: … no record is needed.

> America South Uni: We need both to update current record and then keep a history of that …
>
> Belgium Uni: In our case, things are completely different …

This exchange points to the diversity of institutions present and the extent to which their requirements are similar or, at times, contradictory: where some users require one kind of record to be stored, others need a more comprehensive record, and one institution records things in a different manner altogether! Yet it is here that the Supplier was finally able to observe the similarities and differences between institutions (and to begin to shape them in some way).

These meetings were also interesting for the way in which they appeared to shape the users' attitudes toward the overall generification process and their determination to have particular needs represented in the system. Through spending time getting to know the size and complexity of the task at hand, the participants appeared far more accommodating towards collective requirements, even to the extent that they would often compare institutional practices ('Oh! You do that … '). They had to concede that, even though it was a generic system, the Supplier was determined to search for each and every difference between sites. No differences were ignored. No one group, or so it seemed, was explicitly favoured. Towards the end of one particularly long session some of the users even began to suggest that the SAP was perhaps 'over determined' to find and articulate differences. The America South Uni participant, for instance, described to the others sitting at his table during a coffee break how he thought SAP had 'too much patience' in allowing everyone present to spell out their particularities in such detail.[16] This comment was insightful in that it suggested an interesting shift in the provenance of the generification process and in who takes responsibility for it. Problems were seen to be the result of users, who were intent on describing their particular needs, while the Supplier, who had actually gathered them together in this way, was guilty only of being 'too patient'.

In summary, by shifting design from the level of the individual to that of the community, the Supplier moved the software package from the private domain of each user site, where only particular needs could be articulated, to a public setting, where community or generic requirements could be forged. A further advantage of allowing users to participate collectively was that they were able to 'witness' the continued openness of the process. Indeed, somewhat ironically, some participants expressed concerns that it was not the supplier who was prolonging or complicating the generification process but the users who were doing it to themselves.

Management by Content

Whilst management by community revealed diversity, there was also a need to shape and smooth this diversity; to *manage through content* (Knorr Cetina, 1999).[17] There were two aspects to these strategies: first, to translate collective requirements into functionality that might be used by all of

the sites present; and, second, because these sites were surrogates for potentially *all other* universities, to then translate the community functionality into a much more generic functionality. One method of establishing such templates was through searching for *similarities* between sites. These similarities did not emerge easily, but had to be pursued and actively constructed. Consequently, we think it is useful to describe this process in more detail, and so we focus on a discussion of 'progression' within the CM module.

Process Alignment

One Consultant asks participants to describe their rules for progressing students from one year to another, and to explain how a student's grades contribute to her overall Programme of Study. A complicated conversation develops with various people interjecting. The Consultant struggles to bring the discussion back on topic by attempting to summarize and name the particular process being described:

> SAP Consultant: We've got one aspect now. Just want to get some common things. How [do] we name the baby? Let's go to the grading issue. Want to specify if module will contribute to Programme of Study in any way as a credit or grade. Is there any rule how it contributes? Is it linked to students? What is it linked to that it gives credit?
>
> Swiss Uni: Could be a rule or a decision given by someone?
>
> South African Uni: The student can still do the exam and be graded but it might be true that the grade or credit did or did not influence the student's progression …
>
> Canadian Uni: We wouldn't use these rules: we take all courses into progression. We have rules based on courses students take.
>
> SAP Consultant: It is the same at [America North]. It is the US model. It is the difference between the European and the US model.

There are a number of interesting aspects in this exchange. When faced with diverging requirements, the establishment of generic features seems impossible. However, the Consultant does not admit defeat, but accepts the next best thing to a single generic process: 'two' generic templates. Moreover, she constructs these two templates by aligning or superimposing processes that are already roughly similar to one another ('It is the same at America North'). This then leads to the establishment of a generic feature ('It is the US model'), which means that the requirements of a large group of universities are now seen to have been captured under one process. We also see in this exchange the naming of a further generic template, described as the 'European model', which emerges to capture all the differences that do not fit into the 'US model'. From now on, there will be two modes of progressing students within the CM module (meaning that they will adopt either the US or the European process). Drawing on Epstein (2005) we might describe this as both the production of 'generalized differences' and a form of 'process alignment'. Finally,

once these two categories were established, they were continually compared: both the supplier and the participants acted as if it was self-evident that everything inside each of these processes was identical, and that anything or anyone outside of one classification could be easily accommodated in the other. Indeed, only one of the participants, a South African University, was from an institution outside the USA or Europe. And since interactions during these meetings had shown them that they had many similarities with other users, particularly the British participants, they appeared to be happy to align themselves with the European model.[18] Process alignment appeared to be a successful method, with supplier representatives routinely framing their questions in ways that promoted this form of generification ('Does everyone want the ability to … ?', 'Does anybody else have this?').

Having an Issue Recognized

An interesting, though not altogether surprising, development was that the users began to learn that if they were to have their particular needs represented in the system then they too should engage in alignment work. An America South Uni participant makes a case that the system should record grades for failed courses, and very quickly other users begin to give their support:

> America South Uni: We have concepts called 'forgiveness': a student retakes a course he's not done well in and he is 'forgiven'. The old grade is recorded but not included in the GPA [Grade Point Average].
>
> Canada West Uni: We do the same thing. When we have symbols that aren't graded – like 'withdrawn' or 'incomplete'.
>
> SAP Consultant: This is a big issue for everyone … ?
>
> Canada West Uni: We definitely have to store it. These non-grade things don't have a pass value or fail value, they are a 'third' value.
>
> SAP Consultant: I call it 'additional module results'.

Here, then, an issue is recognized as generic through this accumulation of support. Moreover, the Consultant appears happy to include the feature in the system since she is both able to name it (as 'additional module results') and establish an equivalence among the other institutions whose needs are catered for under this one concept.

The Organizationally Particular

It was common during these sessions to find requests that could not be made compatible across sites. Consequently, they had to be rejected or sifted from the process. The most common method for doing so was simply to categorize requirements as 'specific'. For instance, during a discussion around the storing of surnames, an America East Uni participant describes how they have a specific need to record maiden names after marriage. They suggest adding a new field to the screen (an Info_Type) but the Consultant dismisses this as

unworkable: 'If we went for country-specific or customer-specific Info_Types now, then we could not utilise R/3 resources. The resources would be too great.' On this issue, unlike previous ones, the other universities do not align and thus it is not recorded on the acetate. The official reason for this was that the change would not link back to the generic system (and this meant that CM would no longer integrate with the ERP system of which it was a small part).[19] The suggestion instead is that America East should create a new Info_Type themselves when they customize the module back at their own institution. In other words, making the system fit America East's needs is *postponed* and shifted onto the customization stage at the user site (Hartswood et al., 2002).

Smoothing Strategies

Throughout these requirements-gathering sessions, many of the participants would go into great detail concerning their specific needs. The consultants would often use an interesting range of social strategies and devices to simplify and curtail particular requests, and we explore one such strategy with OHP slides (acetates).

Working the Acetate

In response to one lengthy description, the Consultant used the physical limitations of the acetate to abbreviate a request ('Just trying to think how this can fit all on one line'). On other occasions, particular issues would be rejected for being already covered under existing themes. Pointing to the acetate, the Consultants would say 'we had that issue already', even when it was not always clear just how the new issue had been covered. Indeed the acetate was something of an 'obligatory point of passage' (a device or gateway through which the requirements needed to pass; see Callon, 1986); once scribbled down, an issue could be considered to have been recognized by the Supplier, but, of course, it was far from easy to inscribe it on the acetate. The participants also recognized the importance of the acetates. In one discussion, the university representatives sites are describing progression rules and an America South Uni participant prefaces his intervention by stating that 'you'll need a new page'. While, of course, he is attempting to signal his university's uniqueness, the Consultant dismisses this by pointing to the existing, well-annotated acetate and stating how there is 'one line left'. Later, when the America South participant appears to be about to list a further set of differences, the Consultant states that 'the page is full'. We would say that this *working of the acetate* was a particularly strong form of smoothing because it appeared as a simple material necessity and was thus not recognized as generification work.

From Generification to Generifiers

In the final stages of the CM project there was once again a notable shift concerning the shaping of the package and the locus of generification.

Dragging the design from the private domain of direct user engagement to a public setting had, apparently, been a drain on the Supplier's resources, and the requirements prototyping meetings were no longer seen to be as 'productive' as they once had been. Below, one participant from a Belgium University writes in a report that:

> The current way of working with workshops is very labour intensive for the people of product management and development at SAP Waldorf. The biggest problem is that there is a very mixed public attending these workshops. Some of them already have a lot of expertise in CM and they see the workshops as a roll-in of requirements and for giving feedback after testing. For others this is their first experience with CM and they see it more as a kind of training. SAP wants to change this. In the future there will be standard training courses for larger groups. For roll-in activities there will be focus group meetings. These will only be attended by experts on the subject (limited groups of people) and they will focus on narrow subjects.

This shift was met with objections from users who stated a preference for collective engagement rather than the smaller group or individual interactions. While this appears somewhat counterintuitive, the reason for the objections became clear some weeks later when one user reported that it was now increasingly common for their requests for functionality to be rejected. This was because it was said, by SAP, to be functionality required by only one university. In other words, because there were no longer community meetings, it now appeared difficult for the Supplier to work out, and for the user to determine, what was a generic need and what was not. And it appeared that they had decided to assume that the majority of the requests did not represent generic needs. In order to prove their needs were generic and not particular, the universities had begun to search for similarities between themselves and the other sites (see Pollock & Cornford, 2004). In other words, once back in the private domain, the burden of generification was pushed onto the users. The participants had no choice but to become 'generifiers' themselves. If they did not fully participate in the generification process, if they were not good generifiers, their needs would not be effectively represented within the package. And it appeared to be better to have your needs represented in a generic format than not at all!

Management by Social Authority

The ability of a software package to become mobile is a result of the successful extension of a particularized application, and, at the same time, the extension of the community attached to that system. It is the latter aspect which is of interest – specifically how the process requires the enrolment and configuring of a user community that is subject to, and actively participates in, this generification process. However, the kind of work required in this form of ordering varies from the sophisticated smoothing/sifting strategies and boundary work described above, to what might be described as more direct 'social authority' strategies. This was particularly evident in

later phases of the packages' development when the heterogeneous nature of the user-base and the fact that it was beginning to swell with 'late comers' resulted in pressures to pull the packages in different directions.

Segmenting the User-Base

The initial 'openness' of the package was a useful strategy for building the community by enrolling users into the design process. Now, in the later stages of the package biography, this openness was something of a drawback. As was evident in the quote from the Belgium University above, users were still expecting to have their particular requests met, and what was unsettling some of the established pilots was that the late comers were also making additional demands that might slow or complicate progress. This also occurred in the case of PAMS. The Sales Director describes how early on, when the company did not yet have a finished system, it had had to create an expectation among users that their specific needs would be met. It was now difficult to correct this view:

> … but, of course, it raises a level of expectation … you can be a year downstream in an implementation with somebody, and suddenly they throw up this requirement that has never been vocalised before, but because they bought as an early adopter they perceive that they have that type of relationship that means that you will do it for them. Even though they may well be the only people in the UK that actually want it!

Rather than simply refuse to cater for any kind of particular requirement, however, the Supplier had segmented the community into three distinct categories: as either 'strategic', 'consultative' or 'transactional' customers. While these terms were part of the vernacular of the PAMS team, they were still thought to warrant some explanation by the Managing Director, when he mentioned them to us:

> … it is where we perceive it is worth putting the effort: Strategic Customers, Consultative Customers and Transactional Customers. Transactional customers don't want to spend money. They want everything for nothing. So for every day you put into them you get nothing back. So you put your days into Consultative customers who want to work with and spend with you. Whereas Strategic are all about people who help share the vision of where the product is going to go over the coming years.

From his point of view, Strategic and Consultative customers were central to the future development of PAMS, whereas Transactional customers were peripheral to its evolution. The former were regularly quizzed and consulted on the addition of new features and the general direction of the package while the latter were actively kept at a distance. One example of how this strategy structured the users' interactions with the package was seen in the issue of 'customization' and the question as to whether a user could modify the generic kernel.[20] During a conversation we had with a PAMS programmer, for instance, he praises a modification carried out by

one early adopter and describes how this has even been fed back into the generic package for use at other sites: '[The London Uni] have done a fair bit … 80% of that has been incorporated into the standard package … They were willing to run ahead … they had the resources'. During the same conversation, he criticizes another user for making a modification to the kernel and describes how it was explicitly stated that they are not allowed to make changes to the source code: 'We make sure that it's in the contract that they don't do things like that. We have had customers manipulating the data … from the back-end … Very dangerous … They promised not to do it again.' This suggested that the ability of a user to customize PAMs, and still have their system supported by the Supplier, was directly related to the status they held at that time. This, of course, begs the question as to just how a user might find themselves placed in one or another category.

Good Generifiers

Typically the status of a user was simply related to 'when' they adopted the system: the first group of users being closer to, and later-comers further from, the Supplier. One other key criterion was related to how willing a user was to reshape practices to conform to the templates embodied within the system. The Managing Director of Educational Systems describes how:

> One of the other things we found about Consultative customers where they have entered into a dialogue with us is about how they might change how they do things. There is a lot of functionality in PAMS and there are areas where the universities aren't particularly efficient … So the Consultative customers are more willing to look at how they do their business and how they might improve their business based on suggestions for us based on existing functionality or commissioning us to add extra functionality.

Encouraging users to carry out organizational change to align with the system is an important strategy for managing the user base, and also a way to reduce the need for the further accumulation of particular functionality. It is a method, in other words, of moving users towards the 'organizationally generic'. Moreover, suppliers actively recruit customers who appear willing to engage in such change, and they reward them with greater access to the shaping process.[21]

In summary, Educational Systems does not have the large user base enjoyed by suppliers such as SAP, and thus it has to be sophisticated in how it brings pressure to bear on users. We saw a form of selection where the Supplier was prioritizing which functionality might be allowed into the package. Users were divided into those who sought to align with the organizationally generic features being developed, often through conducting processes of change within their own organizations, and those who did not. The former group, as a reward for being 'good' surrogates, were actively involved in shaping the evolution of the package and were regularly consulted on which features they would like to see in the package. The latter, by contrast, were pushed to the margins of this shaping process, where they
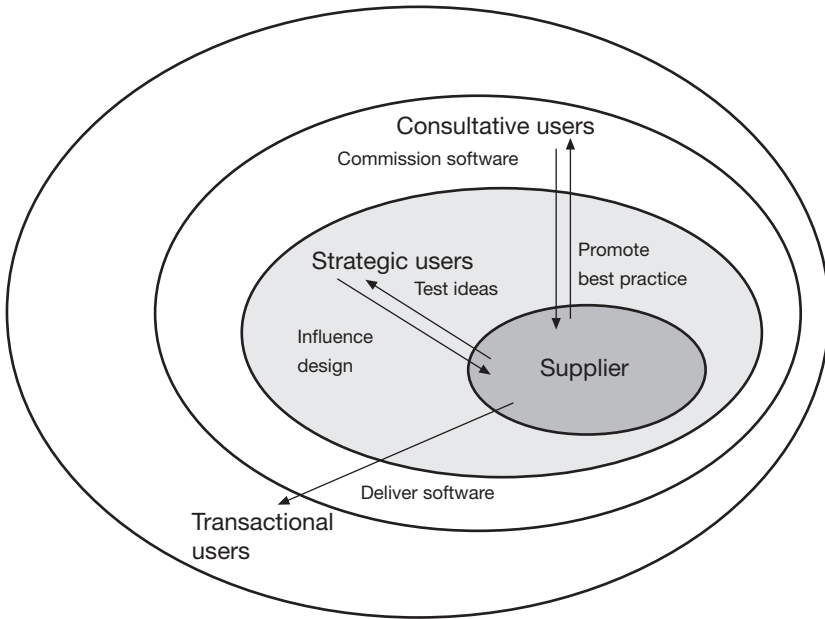
were not consulted or involved in design or evolution. Just what they could do with the system was policed (see Fig. 1).[22]

Promising Future?

We now delineate a final stage of the software packages biography: *the future*. The software packages might be thought to have a promising future or 'career' ahead of them; promising because the effort to create a generic technology required moving towards maturity in order to escape particularization. As a result there are still many places to which the software can travel. In its promotional literature, for instance, SAP boldly states how the CM module embodies 'no country specifics'. Yet, despite what this says, there were times when specific requirements appeared valuable for the circulation of the software package. Or, perhaps, it was simply impossible to avoid including the particular within the generic technologies being built.

Surrogate for Whom?

Some users were able to convince the Suppliers that their needs had 'generic potential'. One criterion determining the ability of a user to get features embodied in the system revolved around the issue of just 'who' they were a surrogate for. The UK market was seen as a 'strong subsidiary'

by SAP, meaning that the inclusion of a British university in the community might open up potential markets elsewhere. And as a result, the British university was able to wield significant influence. For instance, the Supplier agreed to build the 'UCAS admissions link', a piece of functionality that would be a significant drain on resource and, importantly, one that could *not* be applied in other countries. During our research we began to learn that the CM module embodied many other particular features. One document describes how: 'In addition to generic functions, Campus Management also offers country-specific functions. These are functions that are only used in a particular country and cover needs arising from local legislation or business practices.' In other words, including particular functionality allowed the CM module to move within the same sector but also to *different* countries.

The case of Educational Systems raised a different issue, as the addition of particular functionality offered PAMS the potential to move both into a new country and *across* an industrial sector. The Supplier was considering whether to launch PAMS in the USA and, of course, one issue of import was how well PAMS would fit with the peculiarities found there. One area where a difference was perceived was in how student rooms were allocated. Whereas UK students are simply assigned individual rooms, US students typically share a room and can therefore state their preferred type of room mate. The Managing Director described how this difference would require that 'social engineering' software be added to PAMS. Initially sceptical about the costs of such a development, he also saw how this might be useful for the evolution of PAMS:

> That is a piece of functionality that we could add-in and usefully use over here. So it may well be something we can use. One of the things we can certainly use is the ability to have multiple layouts in a room … So we can build those changes into the software in a way that actually positively impacts on our ability to sell the software in the UK.

The addition of this 'social engineering' functionality would mean that PAMS would have more utility in existing UK universities *and* the private sector hotel industry, one area the Supplier had recently targeted. Their aim, in other words, was to identify where particular characteristics could have a more general appeal. We might describe particular features that aid the circulation of the package ('the UCAS admission link', the 'Social Engineering', etc.) as 'generic examples of the particular'.

Paths of Diversity

There were other forms of diversity included in the system. Earlier we discussed the template for the 'progression' of students and how the Consultant had developed not one but 'two' templates. This was interesting as it was one of the rare occasions when the Supplier had to create 'multiple' templates for the same process – what we might describe as *polygeneric* templates. In its promotional literature, the Supplier describes these

poly-generic templates as giving the system extra flexibility through allow-ing adopter more choice:

> Progression – Depending on your particular environment, you may want
> to measure the progress of your students in different ways. One option is
> to determine the academic standing … Another option is to evaluate a stu-
> dent's progress … SAP Campus Management supports several progres-
> sion methods thanks to our global approach to solution design. The
> flexibility of this application allows an institution to change processes in
> the future without the need to install a new student information system.

By allowing poly-generic templates the supplier has created the basis for internally segmenting the user community, so that the templates allow users to follow different routes depending on their particular circum-stances. They have, in other words, established 'paths of diversity' through which users might navigate. This was still a form of generification as the Supplier was allowing users to choose between one of several large group-ings. In this final section we consider what the inclusion of diversity and generality means for shaping the generic system and the community of users.

Opening the Black-Box (and Finding a 'Black-Blob')

We have shown how the generic system results from various kinds of boundary work. With the drawing and redrawing of borders the system embodies a range of features and potentially caters to a wide range of organizations (see Fig. 2).

Let us describe the system. The bulk of its features are the organiza-tionally generic templates that suppliers attempt to build. These form the majority of the organizational 'outer layer', where suppliers hypothesize that organizations are similar and that the participating sites are good sur-rogates for all others in that class of organization.[23] There are also com-promises in which designers, unable to devise a single template, build in several templates to carry out broadly equivalent bundles of organiza-tional processes. These 'poly-generic' features reflect the diversity of user organization practices and contexts that cannot be readily captured within a single template. Finally, there are 'generic particulars', where idiosyncratic requirements are deemed to be important for aiding the future circulation of the package. These are only a few examples of how the generic and the particular are made to fit together. With further research, we would be able to generate further instances and a more com-plex picture. But our point should be clear: when examined closely, generic solutions are not the monolithic systems that much of the litera-ture seems to suppose (see for example Walsham [2001] and Avgerou [2002] as examples). Rather, they are the result of intricate boundary-work involving generification (the creation of generic templates), the par-ticularization of the generic (the poly-generic templates) and, at times,

FIGURE 2
Generic solution as a 'black-blob'
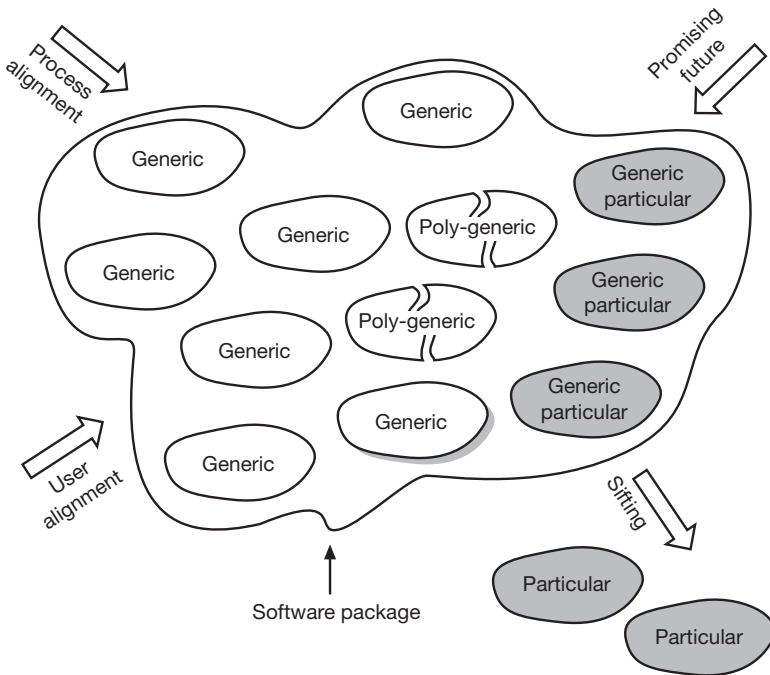
the generification of the particular (the generic particular templates). In our view, the design and evolution of software packages are characterized by the working out of the relationship between the generic and the particular.[24] Indeed, this occurs not simply in design but throughout the lifetime of the software package.

During the research we thus began to recharacterize these generic solution as 'black-blobs' (Michael, 2000). Within STS technologies are commonly described as 'black-boxes' in order to emphasize how their form and function are stable, that prior processes of shaping are obscured, and that the user is configured into using the object in certain ways. By contrast, the software packages are also bounded objects, but their internal workings continually contort as they move around, and as new functionality is added. While the overall appearance of the software package (and in the case of the highly modularized packages such as SAP, as its core 'kernel') may seem to remain intact, the addition of a new template, for example, causes the packages to morph and extend themselves in different directions. It is through this morphing/extension process that software packages are able to move from place to place, and to reach out into new settings. Such amoeboid movements, in turn, enable users to grab onto and then align themselves with the various protuberances and protrusions.

Conclusion: Black-Blobs Travel Better Than Black-Boxes

Certain software packages can be made to travel with 'unique efficiency', to borrow Shapin's (1998) description of scientific knowledge. In doing so, they unsettle prevailing core assumptions in the sociological understanding of organizational technologies. Put simply, much of the sociological and STS literature pays particular attention to the mismatch between system and actual work practices and emphasizes the local adaptation necessary to bridge the gulf (McLaughlin et al., 1999; Walsham, 2001; Avgerou, 2002). While we do not downgrade the importance of this focus on how technologies are imported, we point instead to the need to go beyond studies of 'simple location' and also examine how systems are able to work *across* different organizational contexts and how they are exported. Rather than focus on the collision between unique organizational practices and the generic solution we should *also* address how technologies are made (and continuously remade) to bridge these different locales, as part of our enquiry into the broader and longer-term co-evolution of artefacts and their social settings of use. We have argued that generic solutions *do* exist and that they do travel to many different places; though, of course, they don't go everywhere. They arise through the broader extension of a particularized software application and, at the same time, the management of the user community attached to that solution.

We noted some interrelated moments in the biography of these solutions. There was a distinct *birth stage* at which suppliers designed specific user requirements into the software. This was followed by a number of delimited responses in the subsequent maturation of the package, when the suppliers attempted to move away from the simple accumulation of particular functionality. One interesting aspect was the shift to capture collective rather than individual requirements, in order to establish organizationally generic features through alignment and smoothing practices. Such practices helped establish greater compatibility across sites, as equivalencies were established in organizational practices, and differences were worked together and generalized. Suppliers attempted to align processes that were already roughly similar, what we called 'process alignment work'. The collective gathering of requirements also had a secondary consequence of shifting expectations about the kinds of need that would be met by the system. Through 'witnessing' the level of user diversity, and realizing that the only way to represent needs was to engage in the process, the users' conceptions of their own needs shifted in a way that aligned with those of other participants. In other words, users were in some respects self-governing concerning the articulation of their level of particularity and generality. This raises questions about which users have the capacity to extend and broaden a template: on what grounds and by which methods?

To summarize, it is not just sociologists of science and technology who are interested in the relations between the particular and the generic, and how the boundary between them is established, managed and shifted

(O'Connell, 1993). Software packages are a high-value industrial product, necessitating extensive interactions between suppliers and users. Building software packages calls for suppliers to develop and sustain sophisticated strategies for managing diversity, and setting boundaries and priorities for dealing with their market of user organizations. User organizations similarly need to learn how to respond to and interact with such strategies. As communities grow and inevitably encompass a wider range of organizational types and requirements, this user-base also needs to be organized if the supplier is to avoid being confronted with a potentially overwhelming array of requirements. This, as we have shown, involves different kinds of boundary work – in terms of understandings of which types of organizations lay 'close to' and which 'further from' the supplier's conception of the ideal type of user, and in terms of the willingness of the supplier to accept or sift particular requests from users. The 'black-box' view of the generic solution where it simply 'invades' and 'disciplines' is too crude. What we have shown is that establishing a generic solution is a precarious achievement of various kinds of generification strategies. These are strategies in which the suppliers and users of software packages constantly work towards a pragmatic resolution of the tension between the generic and particular. As a result of this generification work, software packages can circulate and user communities can grow; that is to say, diverse organizations and standard technologies *can* be brought together.

Notes

1. Webster & Williams (1993) report on the difficulties and frequent failures encountered when Computer Aided Production Management (CAPM) systems, designed for large hierarchical US manufacturers, were implemented within the more informal, ad hoc managerial culture and practices of smaller British manufacturers. Fincham et al. (1994) identify similar problems in the transfer of packaged finance service sector administration systems from the USA to the UK where a lower and less formal division of labour prevailed. McLaughlin et al. (1999) discuss the transfer of a hospital management system from one national context to another and suggest that because the system was particular to its geographical birthplace it did not easily translate to new contexts.

2. It has been argued that by the late 1990s most large companies had adopted the same or a similar ERP system (Muscatello et al., 2003). Moreover, these systems are now jumping the boundary from the private to the public sector and are moving into local authorities, hospitals and universities, a move portrayed by many as also highly unlikely.

3. While we do not know of any studies of technology that use this terminology (generification work, the process and attendant strategies of generification), Errington

& Gewertz (2001) provide an interesting discussion of generification in terms of the local culture of indigenous peoples and how it is affected by other, more dominant forms of knowledge. We work up the notion of generification because we think it indicates a way of making sense of how software packages are developed and recycled, and also provides a counter to biases towards localization arguments within current STS.

4. See also Hales (1994) for this view.

5. In their comparative study of IT systems, to give just one compelling example, McLaughlin et al. (1999) deploy a commonplace vocabulary to highlight how users actively 'appropriate' (MacKay & Gillespie, 1992), 'domesticate' (Sorensen, 1996) or 'work-around' (Gasser, 1986) the shortcomings of newly arrived technologies.

6. An exemplary instance of this kind of writing is Avgerou's (2002) recent book.

7. The concept of narrative bias invites us to reflect upon the repertoires of classic stories that particular schools of analysis often develop with characteristic contexts, problem diagnosis, dangers and solutions (Williams et al., 2005). See also Woolgar & Cooper (1999) for a similar discussion of 'iconic exemplars' in STS.

8. Thanks to Michael Lynch for framing this point in this way.

9. We are grateful here to Jamie Fleck for bringing this set of arguments to our attention.

10. We should also mention Timmermans & Berg's (1997) work as they have suggested that artefacts can be both universal and local at the same time. Putting forward the notion of the 'local universal', they argue that universals *do* exist but they merge together with the local. This is an important contribution. However, our interests are different in some respects. Their account is firmly on the side of work practice and the appropriation of a medical standard and how despite various 'local circumventions' and 'repairs' carried out by users of a particular protocol, the notion of 'one' standard still persists. Also, *local universal* is an analytical notion they invent to separate out the world of practice from the world of standards, and then to show how these worlds are reconciled with one another. Our concerns, in contrast, are with design practices and how actors *themselves* negotiate and establish the boundaries between what is particular and generic. And in this regard we view as sociologically interesting the way *suppliers* attempt to bring together and manage both of these aspects while building a generic software package. Gieryn (1999) discusses a similar point in relation to the authority of science and how lay people understand what counts as good and bad science. It is important, he says, to focus on how actors perform this boundary-work rather than privileging the analysts' view.

11. For a more detailed discussion of the 'biography' of a software package see Pollock et al. (2003).

12. The material presented here stems from observations (by N.P.) of what are sometimes called 'requirements prototyping' sessions (meetings in which suppliers demonstrate early versions of systems and elicit feedback), and user group meetings at the suppliers' premises. A number of semi-structured interviews and informal discussions were also conducted with supplier consultants, programmers and users. Finally, one of the authors (N.P.) was commissioned to conduct a study on the suitability of launching PAMS abroad. Along with a co-researcher, Tasos Karadedos, N.P. met regularly with the management team to discuss strategies and potential markets. Material from this study is also presented here.

13. This discussion of Accumulative Functionality is partially drawn from Karadedos (2003).

14. Here we loosely draw on Woolgar's (1996) notion that a technology 'performs' a community. He uses the term in conjunction with the 'technology as text' metaphor to show how readers arrive at a preferred form of use. He suggests that within the technology/text certain identities and positions are offered with which the user can choose to align.

15. This was taken from an email exchange between one of the pilot sites and the supplier. The author was discussing the danger of design that was focused on individual sites and not the community.

16. Indeed the participants were becoming increasingly frustrated by the supplier's attempts to understand each and every difference among all the universities present and to reconcile these with the needs of the others present. For the suppliers, such a process appeared to be useful, as they saw it as a means by which the module might become *more generic* and thus potentially applicable to the widest variety of higher education institutions.

17. Knorr Cetina develops the notion of 'management by content' to describe how people are managed specially through the content of their work as opposed to management through organizational structure or hierarchy (1999: 172).

18. We later found out during the final stages of drafting this paper that the South African University eventually decided not to implement CM. Their reasons, and the continuing evolution of CM, are the subject of continuing research.

19. There is an interesting issue here of how the universities were squeezed into existing software models that had nothing to do with higher education. We have explored this issue in Pollock & Cornford (2004).

20. Usually changes to the source code provide suppliers with something of a dilemma. On the one hand, modifications developed by users are an important source of innovation and are often fed back into the generic package for use at other sites. On the other hand, such evolution can be disruptive and if things go wrong during such modifications, this often leads to disputes about where responsibility rests for sorting things out. See Pollock (2005) for a lengthy discussion of this issue in relation to the authorized and unauthorized customizations and 'work-arounds' conducted on standardized computer systems.

21. Interestingly, we also routinely witnessed how a user might shift from one classification to another. The very first adopter of PAMS, for instance, was in the process of moving from the centre to the periphery (and there was even talk that it was now becoming 'transactional').

22. This diagram is a development of one found in Karadedos (2003). Permission to reproduce it has been granted.

23. These are of course equivalences only in the *realm of design* and whether they emerge in the *realm of practice* will depend on other generification strategies.

24. Indeed the globalization theorist Roland Robertson (1992: 102) has gone as far as to describe 'contemporary globalisation' as marked by a similar process or what he describes as the ' … institutionalisation of the two-fold process involving the universalisation of particularism and the particularisation of universalisation'.

References

Avgerou, Chrisanthi (2002) *Information Systems and Diversity* (Oxford: Oxford University Press).

Bansler, Jorgen & Erling Havn (1996) 'Industrialised Information Systems Development', *CTI Working Paper* No 22. Technical University of Denmark.

Berg, Marc (1997) *Rationalizing Medical Work: Decision-Support Techniques and Medical Practices* (Cambridge, MA: MIT Press).

Brady, Tim, Margaret Tierney & Robin Williams (1992) 'The Commodification of Industry Applications Software', *Industrial and Corporate Change* 1(3): 489–514.

Callon, Michel (1986) 'The Sociology of an Actor-Network: The Case of the Electric Vehicle', in Michel Callon, John Law & Arie Rip (eds), *Mapping the Dynamics of Science and Technology* (London: Macmillan): 19–34.

Cambrosio, Alberto & Peter Keating (1995) *Exquisite Specificity: The Monoclonal Antibody Revolution* (New York: Oxford University Press).

Cooper, Robert (1998) 'Assemblage Notes', in Robert Chia (ed.), *Organized Worlds: Explorations in Technology and Organization with Robert Cooper* (London: Routledge): 108–29.

Davenport, Thomas (2000) *Mission Critical: Realising the Promise of Enterprise Systems* (Boston, MA: Harvard Business School Press).

De Laet, Marianne & Annemarie Mol (2000) 'The Zimbabwe Bush Pump: Mechanics of a Fluid Technology', *Social Studies of Science* 30(2): 225–63.

Epstein, Steven (2005) 'Institutionalizing the New Politics of Difference in U.S. Biomedical Research: Thinking across the Science/State/Society Divides', in Scott Frickel & Kelly Moore (eds), *The New Political Sociology of Science: Institutions, Networks and Power* (Madison, WI: University of Wisconsin Press).

Errington, Frederick & Deborah Gewertz (2001) 'On the Generification of Culture: From Blow Fish to Melanesian', *Journal of the Royal Anthropological Institute* 7(3): 509–25.

Fincham, Robin, Jamie Fleck, Rob Procter, Harry Scarbrough, Margaret Tierney & Robin Williams (1994) *Expertise and Innovation* (Oxford: Clarendon Press).

Friedman, Andrew & Dominic Cornford (1989) *Computer Systems Development: History, Organization and Implementation* (Chichester: John Wiley).

Gasser, Les (1986) 'The Integration of Computing and Routine Work', *ACM Transactions on Office Information Systems* 4(3): 205–25.

Gieryn, Thomas (1999) *Cultural Boundaries of Science: Credibility on the Line* (Chicago, IL: University of Chicago Press).

Hales, Mike (1994) 'Where Are Designers? Styles of Design Practice, Objects of Design and Views of Users in CSCW', in Duska Rosenberg & Chris Hutchinson (eds), *Design Issues in CSCW* (London: Springer-Verlag): 151–78.

Hanseth, Ole & Kristine Braa (2001) 'Hunting for the Treasure at the End of the Rainbow: Standardising Corporate IT Infrastructure', *Computer Supported Cooperative Work (CSCW)* 10: 261–92.

Hartswood, Mark, Rob Procter, Roger Slack, Alex Voss, Monica Buscher, Mark Rouncefield & Philippe Rouchy (2002) 'Towards a Principled Synthesis of Ethnomethodology and Participatory Design', *Scandinavian Journal of Information Systems* 14 (2): 9–30.

Karadedos, Tasos (2003) *The Biography of a Software Package*, Unpublished MSc Dissertation, University of Edinburgh.

Knorr Cetina, Karin (1981) *The Manufacture of Knowledge: An Essay on the Constructivist and Contextual Nature of Science* (Oxford: Pergamon Press).

Knorr Cetina, Karin (1999) *Epistemic Cultures: How the Sciences Make Knowledge* (Cambridge, MA: Harvard University Press).

Knorr Cetina, Karin & Klaus Amann (1990) 'Image Dissection in Natural Scientific Inquiry', *Science, Technology, & Human Values* 15(3): 259–83.

Latour, Bruno (1987) *Science in Action: How to Follow Scientists and Engineers Through Society* (Cambridge, MA: Harvard University Press).

Latour, Bruno (1999) *Pandora's Hope: Essays on the Reality of Science Studies* (Cambridge, MA: Harvard University Press).

MacKay, Hugh & Gareth Gillespie (1992) 'Extending the Social Shaping of Technology Approach: Ideology and Appropriation', *Social Studies of Science* 22: 685–716.

McLaughlin, Janice, Paul Rosen, David Skinner & Andrew Webster (1999) *Valuing Technology: Organizations, Culture and Change* (London: Routledge).

Michael, Mike (2000) *Reconnecting Culture, Technology and Nature* (London: Routledge).

Muscatello, Joseph, Michael Small & Injazz Chen (2003) 'Implementing Enterprise Resource Planning (ERP) Systems in Small and Midsize Manufacturing Firms', *International Journal of Operations and Production Management* 23(7/8): 850–66.

O'Connell, Joseph (1993) 'Metrology: The Creation of Universality by the Circulation of Particulars', *Social Studies of Science* 23: 129–73.

Ophir, Adi & Steven Shapin (1991) 'The Place of Knowledge: A Methodological Survey', *Science in Context* 4: 3–21.

Pollock, Neil (2005) 'When is a Work-around? Conflict and Struggle in Computer Systems Development', *Science, Technology, & Human Values* 30(4): 496–514.

Pollock, Neil & James Cornford (2004) 'ERP Systems and the University as a "Unique Organization"', *Information Technology and People* 17(1): 31–52.

Pollock, Neil, Robin Williams & Rob Procter (2003) 'Fitting Standard Software Packages to Non-Standard Organizations: The Biography of an Enterprise-wide System', *Technology Analysis and Strategic Management* 15(3): 317–32.

Quintas, Paul (1994) 'Programmed Innovation? Trajectories of Change in Software Development', *Information Technology and People* 7(1): 25–47.

Ravetz, Jerome (1972) *Scientific Knowledge and its Social Problems* (Oxford: Oxford University Press).

Robertson, Roland (1992) *Globalisation; Social Theory and Global Culture* (London: Sage Publications).

Rolland, Knut & Eric Monteiro (2002) 'Balancing the Local and the Global in Infrastructural Information Systems', *Information Society* 18(2): 87–100.

Salzman, Harold & Stephen Rosenthal (1994) *Software by Design: Shaping Technology and the Workplace* (Oxford: Oxford University Press).

Sawyer, Steve (2000) 'Packaged Software: Implications of the Differences from Custom Approaches to Software Development', *European Journal of Information Systems* 9: 47–58.

Sawyer, Steve (2001) 'A Market-Based Perspective on Information Systems Development', *Communications of the ACM* (November) 44(11): 97–101.

Shapin, Steven (1998) 'Placing the View From Nowhere: Historical and Sociological Problems in the Location of Science', *Transactions of the Institute of British Geographers* 23(1): 5–12.

Sorensen, Knut (1996) 'Learning Technology, Constructing Culture: Socio-technical Change as Social Learning', *STS Working Paper* No. 18/96, Centre for Technology & Society, Trondheim, Norway.

Star, Susan Leigh & Karen Ruhleder (1996) 'Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces', *Information Systems Research* 7(1): 111–34.

Suchman, Lucy (1994) 'Working Relations of Technology Production and Use', *Computer Supported Cooperative Work (CSCW)* 2(30): 21–39.

Timmermans, Stefan & Marc Berg (1997) 'Standardization in Action: Achieving Local Universality Through Medical Protocols', *Social Studies of Science* 27: 273–305.

Turnbull, David (2000) *Masons, Tricksters and Cartographers* (London: Routledge).

Walsham, Geoff (2001) *Making a World of Difference: IT in a Global Context* (Chichester: Wiley).

Webster, Juliet & Robin Williams (1993) 'Mismatch and Tension: Standard Packages and Non-standard Users', in Paul Quintas (ed.), *Social Dimensions of Systems Engineering* (Hemel Hempstead: Ellis Horwood): 179–96.

Whitehead, Alfred North (1967) *Science and the Modern World* (New York: The Free Press).

Williams, Robin, James Stewart & Roger Slack (2005) *Experimenting with Information and Communication Technologies: Social Learning in Technological Innovation* (Cheltenham: Edward Elgar).

Woolgar, Steve (1996) 'Technologies as Cultural Artefacts', in William Dutton (ed.), *Information and Communication Technologies: Visions and Realities* (Oxford: Oxford University Press): 87–102.

Woolgar, Steve & Geof Cooper (1999) 'Do Artefacts Have Ambivalence: Moses' Bridges, Winner's Bridges and Other Urban Legends in STS', *Social Studies of Science* 29(3): 433–49.

Neil Pollock lectures in the sociology of information and communication technologies (ICTs) at the University of Edinburgh and is a member of the Research Centre for Social Sciences. He has co-authored (with James Cornford) *Putting the University Online: Information, Technology and Organizational Change* (Open University Press, 2003) and is currently writing

a book (with Robin Williams) for Routledge entitled *Shaping Software for Organizations: The Biography and Social Life of a Software Package*.

Address: The University of Edinburgh, School of Management, George Square, Edinburgh, EH8 9JY, Scotland, UK; fax: +44 131 6683053; email: neil.pollock@ed.ac.uk

Robin Williams is Professor of Social Research on Technology and Director of the Research Centre for Social Sciences, University of Edinburgh. He has published widely on the social shaping of ICTs and has written (with James Stewart and Roger Slack) *Experimenting with Information and Communication Technologies: Social Learning in Technological Innovation* (Edward Elgar, 2005).

Address: ESRC Innogen Centre, The University of Edinburgh, Old SurgeonsHall, High School Yards, Edinburgh EH1 1LZ, Scotland, UK; fax: +44 131 650 6399; email: r.williams@ed.ac.uk

Luciana D'Adderio is Senior Research Fellow at the Research Centre for Social Sciences, University of Edinburgh. She has recently published *Inside the Virtual Product: How Organizations Create Knowledge through Software* (Edward Elgar, 2004) and is currently writing a book on the 'Organization of Testing'.

Address: Research Centre for Social Sciences, The University of Edinburgh, Old Surgeons Hall, High School Yards, Edinburgh, EH1 1L2, Scotland, UK; fax: +44 131 650 6399; email: luciana@inf.ed.ac.uk