Infrastructures From the Bottom-Up and the Top-Down: Can They Meet in the Middle?

Michael B. Twidale
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
501 E. Daniel St. Champaign, IL 61820 USA
twidale@uiuc.edu

Ingbert Floyd
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
501 E. Daniel St. Champaign, IL 61820 USA
ifloyd2@gmail.com

ABSTRACT
Based on a study of participatory design in the development of cyberinfrastructure involving the rapid composition of open source software and web services, we consider cases where researchers create their own ad hoc infrastructures out of available software. We compare 'top-down' and 'bottom-up' cyberinfrastructure development and speculate on whether the two approaches can be productively combined.

Keywords
Open Source Software, Cyberinfrastructure, Research Collaboratories, End-User Computing, Appropriation.

INTRODUCTION
Participatory design (PD) originally focused on novel workplace information system development, typically a top-down managerial initiative. This focus was natural given that when PD first emerged as a method, most contexts of interest were attempts at initial computerization of a particular work setting. Hence design typically involved building applications from scratch and integrating them into existing sociotechnical systems.

Current workplace information ecologies in most developed countries no longer resemble those early environments. Not only has computing technology become widespread, but certain applications such as word processors, spreadsheets, and web browsers have become practically ubiquitous in many workplaces. Recent trends in application development such as the free/libre open-source software (FLOSS) movement and the proliferation of web applications and web APIs continue to alter these ecologies, and change the expectations of people within them. While constructing new, large-scale software applications is still necessary, our on-going research is uncovering a number of emergent design practices by both amateurs and professionals which are remarkably participatory and user-driven in nature, and exist only as a result of how current information ecologies are developing.

We support Robertson's argument [12] that PD research should go beyond novel system development and study everyday design practices such as adoption and tailoring. Early PD work addressed challenges of appropriate design under conditions of scarcity – how to get from no software to some software that was actually useful. Now we have the same challenge of appropriate design, but under conditions of (partial) abundance – lots of applications and services that can be selected and combined to get at least near to what is needed.

In this paper we consider academic research collaboration, and how various pre-existing computational technologies can be assembled to support and transform work there. We note that similar emergent design activities are occurring both in top-down, government-funded collaboratories and cyber-infrastructures, and in bottom-up, unfunded research collaboration, as well as assemblies of technologies occurring in parallel to 'official' cyberinfrastructures. Thus we have an opportunity for comparison between the two types of activity, and an opportunity to investigate how lessons learned from one activity might benefit the other.

PATCHWORK PROTOTYPING
We have been involved (as both observers and participants) with several groups developing collaborative systems. In each group we noted ad hoc prototyping and development strategies that emerged somewhat independent of each other but were remarkably similar. Subsequently, we have collected anecdotal evidence of similar approaches being used elsewhere, both in academic settings but also in commercial in-house software development.

We call this approach patchwork prototyping (see [5, 6] for more detail). It is an example of the kind of emergent PD activity that current information ecologies enable. It involves using combinations of FLOSS, web services, mash-ups, and locally developed code in order to create, test and rapidly iterate high-fidelity prototypes which users can integrate into their daily work activity, and is thus true "design in use" [4]. It has three major components:

- Rapid iteration of high-fidelity prototypes;

- Incorporation of the prototypes by the end users into their daily work activities;
- Extensive collection of feedback facilitated by an insider to the user community.

When integrated, these components enable developers to access and respond to users' needs while those needs are evolving. In the process, the ever-changing prototype serves as a mediator for the articulation work that needs to occur for the users to establish new work practices.

For example, one group (consisting of system developers and representatives of the intended users, chiefly academic faculty and graduate students) was building a cyberinfrastructure for environmental engineers. The team was not merely trying to articulate their requirements, but also to understand what the technology could provide in terms of collaborative support and data sharing and how this might change the way that they did their research. As such, the design process was necessarily exploratory. The evolving system was used as part of the planning and management processes of the project, necessarily involving the sharing of various documents, and so being a more authentic test of both the software and what was needed.

Patchwork prototyping is a type of cooperative prototyping [1, 8]; however it blends the design and implementation phases of the development process, because the prototype is incorporated almost immediately into users' everyday activities, and because production-scale modules can gradually be introduced as they are developed to replace the FLOSS applications used as prototypes to uncover the requirements. The method has five stages, and an iteration normally takes no longer than a week:

1. Make an educated guess about what the target system might look like;
2. Select tools which support some aspect of the desired functionality;
3. Integrate the tools into a rough composite;
4. Deploy the prototype, solicit feedback from users;
5. Reflect on the experience of prototype building and on the user feedback, and repeat - quickly.

Patchwork prototyping works well for ill-defined situations where neither the developers nor the users have a clear idea of what they need the software to do, but rather have an idealized vision of the kinds of things computing technology might enable users to accomplish. Prototypes can be radically altered by adding or removing FLOSS components, changing default configurations, or by reconfiguring the interface. Most patchwork prototypes are web-based, making interface reconfigurations fast and easy. Such rapid and visible change ensures that users do not fixate on a particular design because they are presented with a new version before they have time to grow comfortable with its idiosyncrasies. This allows both designers and users to explore a larger design space, helping users develop a more concrete understanding of what is possible with the technology, enabling them to make better design recommendations. Furthermore, we have observed that if users have problems with a particular feature (e.g., a wiki or forum system), a different implementation can rapidly be substituted; giving users an opportunity to test whether their distaste is an issue surrounding the particular interface or functionality, or the whole idea.

Patchwork prototyping requires FLOSS. The ability to modify the source code is vital to effective integration of the modules, thus precluding the use of Commercial Off-The-Shelf (COTS) software [2]. It seems the power of patchwork prototyping to overcome common barriers to successful information system design (e.g. [9, 10]) is a direct result of current and emerging information ecologies.

Patchwork prototyping can be seen as an application of many of the methods developed in PD, but exploiting the possibilities of pre-existing software. It can be compared to other rapid prototyping and development techniques including paper prototyping. Due to its use of pre-existing software, it is very fast, but still results in a working usable (and testable) system. It also seems to support discussions with end-users – in part because they may be familiar with some of the existing applications, or can immediately try them as deployed in authentic activities. This helps discussions about creating new requirements for an envisaged system that will involve combinations of functionalities in those applications, but often in new, interesting ways with additional tailoring and supplementary functionality. As such it emulates some of the creativity embodied in the design of mashups by expert programmers (combining existing functionalities and interfaces in new ways), but in a manner that does not require computational expertise.

We are not claiming to have invented or refined a new PD technique. Rather we are noting a phenomenon that we believe to be widespread – that the availability of existing software is allowing much more design by composition, instead of solely design from scratch. Of course, patchwork prototyping has some significant limitations. The obvious one is that it needs appropriate existing code or web services to create the prototype quickly. It also requires skilled and sensitive developers, and significant leadership and feedback collection by user-group leaders and insiders in order to have rapid and effective iteration cycles. Finally, all the projects where we observed patchwork prototyping had a decent amount of funding to pay developers and maintain the computing infrastructure.

COMPOSED SOCIOTECHNICAL INFRASTRUCTURES

Recently, we have been informally considering how researchers manage to collaborate even if they have no or minimal funding to explicitly support this. Our purpose is to study the process by which both amateur and professional designers engage in the process of composing

collaboration infrastructures out of at-hand or otherwise easily available applications or services. It should be noted that this is in the context of an overall well-funded research university. This is not a study of resource poverty. Rather, it is the study of how in a setting of widespread access to computers, technology, bandwidth, and skills it is possible to put together and tailor workable collaborative systems with little additional support.

This is not necessarily a matter of building an integrated cyberinfrastructure using components, but can simply involve downloading or using a combination of applications and web services, manually copying data between them in order to get the job done in an ad hoc but fast and low cost way. It can be as simple as working on a distributed project using a combination of email, spreadsheets, Google docs, Skype, Yahoo groups, Flickr, and various personal and public calendars. These services are typically sufficiently lightweight that it is easy both to assemble them and to try out and integrate new services, keeping them and replacing an older one or rejecting and reverting as needed. With small groups the process of trialing and switching is so fast that it seems to be unnecessary to do traditional requirements capture and assessment activities. Unlike patchwork prototyping, this use often involves COTS software. This is harder to integrate seamlessly into an overall designed application, but files can still be integrated even if simply by manual online sharing and copy-paste, gaining the advantages of familiarity and relatively low cost, without requiring substantial technical knowledge – a kind of bricolage [3].

Existing work on technological selection, adoption, adaptation, tailoring, appropriation and innovation can inform this analysis. However, that work normally focuses on a single, integrated application (e.g., [4]), and here the whole point is that there are multiple applications, and more available all the time to be composed or replaced – a kind of artful integration [13].

While our research is still in progress, we believe this activity bears some resemblance to patchwork prototyping, but that it is often severely constrained by a lack of resources. For example, we have encountered several cases of researchers who have knowledge of fields such as CSCW, PD, and HCI, who have engaged with system design and development, but who have settled for infrastructures that were minimally useful because they had limited access to server space, the access they did get took considerable time to negotiate (or they simply gave up and used free web services), they don't have time to perform maintenance activities on the infrastructure themselves, and they don't have funds to hire someone to perform the maintenance activities for them. As a result, even though they recognize the need for requirements gathering, prototyping, and iteration of designs, they had no ability to engage in such activity, and felt that some infrastructure to support their tasks was better than no infrastructure.

Other people we have observed have created very complex collaboration infrastructures by creatively integrating software packages into a community workflow. However, these infrastructures are often unstable, as the resources they utilize are often temporary in nature. Thus, the users are constantly migrating software platforms, services, and collaboration spaces. Such arrangements work for small-scale and short-term projects that can be completed, and whose product can be stored on more stable infrastructures. While it would seem that such a solution would not always work so well in supporting sustained, long-term collaboration, the collaborative activity we have observed so far has outlasted several changes in infrastructure, suggesting that as long as some aspect of the infrastructure remains relatively stable (file storage space, web-service email storage, etc.), changing infrastructures can be worked around as long as the groups are small enough and long-term relationships between the group members have been established. However, this does not solve the problem of how to enable researchers to collaborate who are interested in working together but have no history of collaboration, and thus still need to work out shared practices, vocabulary, standards, and compatible values.

BOTTOM-UP VS. TOP-DOWN COLLABORATORIES

The funding model for collaboratories, cyberinfrastructures and related resources (such as e-science and e-social science in the UK) follow a model that we would call 'top-down'. That is, a resource is funded centrally (usually from a research foundation such as NSF or JISC) that will be of use to a reasonably large, distributed research community enabling both greater collaboration and the sharing and use of scarce equipment, technical resources and high end computing power. The approach may or may not use PD techniques to support the activity. PD may be considered unnecessary as the main participant stakeholders – the researchers themselves – are central to writing and obtaining the grant. The original work on patchwork prototyping occurred in such top-down settings.

However, many researchers are interested in collaborating on projects for which there is limited or no grant funding to support. In many such cases, only a minimal infrastructure is needed to support their research. As a result they are unable to make the case that the work involves innovative, indeed glamorous, computing work.

As a result of these two factors, many potentially fruitful collaborations go unrealized. We believe that the lightweight ad hoc methods of creating infrastructures by composition outlined in the previous section might be a productive solution – a indigenous bottom-up approach to development using locally available resources – and one having intriguing similarities with appropriate technology work in development studies.

In addition to bottom-up prototyping that can help poorly resourced projects, we suspect that considerable bottom-up activity also occurs amongst researchers in funded top-

down cyberinfrastructure settings. To date we only have anecdotal evidence for this, but it seems that the recurrent practice of workarounds and the abundance of and ease of using web services allows for the emergence of a 'shadow cyberinfrastructure', bypassing the main one in cases where time, convenience or necessity mean that the official system does not quite do what is needed and a grubby combination of applications is good enough for the job.

If permitted, bottom-up activities amongst teams can also support innovation in a top-down project. Karasti & Syrjänen's "cherry-picking octopus" [7] is a powerful example of how heterogeneity in approaches at the site level allows experimentation and "prototyping into consensus".

The bottom-up approach can never provide the high-end functionalities that the top-down approach promises. However it is fast, and very robust under changing needs and opportunities. It remains to be seen whether the bottom-up and top-down infrastructures are necessarily in competition with each other. Are bottom-up approaches simply symptoms of the inflexibility or slowness to completion of the top-down designed system? Are they the bazaar that springs up alongside while the cathedral is laboriously constructed over centuries [11]? Are they just an accumulation of workarounds by idiosyncratic nonconformists, a kind of black market in officially unsanctioned and unsupported collaborative technologies that system administrators are unable or unwilling to incorporate into the official infrastructure?

Or can top-down and bottom-up methods meet in the middle, combining the strengths of each in a truly robust, powerful, adaptable and flexible infrastructure? Can the bottom-up approach create an exploratory testbed of ideas that can feed requirements into the larger systematic development activities of the top-down approach? Can the techniques of patchwork prototyping that require skilled system developers to implement be integrated with the less powerful but faster techniques of assembling and tailoring applications? Can the PD-inspired approach of the former fit with the open innovation [14] approach of the latter? What kind of robust base-infrastructure (servers, permissions, archiving, etc.) can support bottom-up innovation in creating the next tier of infrastructure? We are not sure, but acknowledging the existence of, bottom-up methods and then taking steps to understand how they operate in various contexts of rich and poor resources seems a good place to begin.

CONCLUSION

By using available applications, code and web services it is possible to support both top-down cyberinfrastructure development using traditional PD techniques to enable rapid development testing and reflection on use, as well as bottom-up cyberinfrastructure development where end users assemble resources to create a lightweight ad hoc environment to support collaborative interaction. It remains to be seen if these two approaches are necessarily in competition or if they can be productively combined.

REFERENCES

1. Bødker, S. & Grønbæk, K. (1991) Cooperative Prototyping: Users and Designers in Mutual Activity. *International Journal of Man-Machine Studies*, 34(3) 453-478.
2. Boehm, B.W. & Abts, C. (1999) COTS Integration: Plug and Pray? *IEEE Computer*, 32(1) 135-138.
3. Büscher, M., Gill, S., Mogensen, P. & Shapiro, D. (2001) Landscapes of Practice: Bricolage as a Method for Situated Design. *Computer Supported Cooperative Work* 10(1) 1-28.
4. Dittrich, Y., Eriksen, S., & Hansson, C. (2002) PD in the Wild; Evolving Practices of Design in Use. *Proceedings of PDC 2002*. 124-134
5. Floyd, I.R., Jones, M.C., Rathi, D. & Twidale, M.B (2007) Web Mash-ups and Patchwork Prototyping: User-driven technological innovation with Web 2.0 and Open Source Software. *Proceedings of HICSS 2007.*
6. Jones, M.C., Floyd, I.R., Twidale, M.B. (2007) Patchwork Prototyping with Open-Source Software. In St. Amant, K. & Still, B. (Eds), *The Handbook of Research on Open Source Software* Idea Group, Inc.
7. Karasti, H. & Syrjänen, A-L. (2004) Artful infrastructuring in two cases of community PD. *Proceedings of PDC 2004*, 20 – 30.
8. Kensing, F., & Blomberg, J. (1998) Participatory Design: Issues and Concerns. *Computer Supported Cooperative Work*, 7(3-4) 167-185.
9. Luke, R., Clement, A., Terada, R., Bortolussi, D., Booth, C., Brooks, D., & Christ, D. The Promise and Perils of a Participatory Approach to Developing an Open Source Community Learning Network. *Proceedings of PDC 2004*, 11-19.
10. Pekkola, S., Kaarilahti, N., & Pohjola, P. (2006) Towards Formalised End-User Participation in Information Systems Development Process: Bridging the Gap between Participatory Design and ISD Methodologies. *Proceedings of PDC 2006*. 21-30.
11. Raymond, E.S. (1999) The Cathedral and the Bazaar, Sebastopol, CA: O'Reilly Press.
12. Robertson, T. (1998) Shoppers and Tailors: Participative Practices in Small Australian Design Companies. *Computer Supported Cooperative Work* 7, 205-221.
13. Suchman, L. (2002) Located Accountabilities in Technology Production. *Scandinavian Journal of Information Systems* 14(2) 91-105.
14. von Hippel, E. (2005) *Democratizing Innovation.* Cambridge, MA: MIT Press.