# Design Informatics – Information Needs in Design

**David G. Hendry**
Information School
University of Washington Seattle, WA 98195
dhendry@u.washington.edu

**ABSTRACT**
The position I would like to advance in this workshop is that a deeper understanding for design can be obtained by taking an *information perspective* on design activities. Under this perspective the major unit of analysis is the information transaction – the specific needs and tasks associated with capturing, storing, updating, linking, and accessing information. By focusing on the information capacities that design teams create for themselves and by describing them with a technologically-neutral vocabulary, we can begin to recognize commonalities that span design methodologies. This approach offers a strategy for developing a more unified view of design which, in turn, can provide insight into the requirements of design information systems and elucidate new areas of design competency and opportunity.

**Author Keywords**
Design, Information Needs, Design Information Systems, Design Informatics

## INTRODUCTION

Design is information intensive. In 1965, for example, Archer [1] introduced a normative, stage model of design with "data collection" at its center (Figure 1). The model shows the interpenetration or cross-connectedness of design activities and information-handling activities such as capturing relevant information, recording information in documents, organizing documents, finding documents, and seeking information from experts. When Archer introduced this model, he seemed to assume that the demands of the design process would cause information handling to unfold in a straightforward fashion.

In any case, his writing does not discuss the problematic connection between "design," that is, furthering the thing that is to exist, and "documenting," that is, recording what has been asserted or discovered about the thing. Indeed, a fundamental trade-off in all settings of design is that if time is spent documenting for uncertain future benefits, it is taken away from designing for immediate progress [8].

The costs associated with documenting can be divided into two components: 1) *Cost of knowledge* [4], that is, the costs associated with finding some information in a particular kind of system; and 2) *Cost of update* [7], that is, the costs associated with adding, updating, deleting, information that might be needed in the future. A significant long-term challenge for design, especially in the Participation Age [16], is developing an understanding for how best to manage these costs and how to weigh them against the present and future benefits of the collected information.

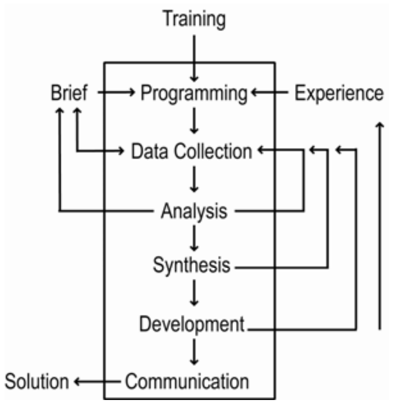

**Figure 1. Information collection and flow in design (Redrawn from [1].)**

## DESIGN INFORMATION SYSTEMS

Design information systems, as I shall call them, are physical or electronic systems that endow teams with particular capacities for design. Because they are information systems, they also entail the costs of knowledge and update. To demonstrate these costs, consider four examples.

First, the IDEO TechBox [11] is a cabinet containing an eclectic collection of artifacts that illustrate new materials or recent innovations that can be touched and exaimed. The TechBox is intended to promote analogical reasoning and creative exploration. Because of its relatively small size, central location, and loose organization of artifacts both the costs of knowledge and update are low.

Second, the Manifesto for Agile Software Development's [3] second principle says to minimize formal documentation in favor of working systems. Consultants who teach Agile techniques often begin by eschewing electronic tools, even simple ones. Then, they introduce the use of PostIt Notes on a centrally located whiteboard, guiding the team to track very simple information – for example, features being implemented, features implemented but not tested, and features still to be prioritized, and so on. With the use of color and spatial groupings such information is readily tracked. Like the TechBox but for a different purpose, this design information system minimizes both the cost of knowledge and update; further, the system provides the team with situational awareness of the overall state of the project.

The downside of the artful use of PostIts and a whiteboard, however, is that it works for collocated teams only. To enable participation by developers who are physically remote, the team might agree to take images of the whiteboard at regular intervals and to post them on, for example, a Wiki. This move, on the one hand, provides a new capacity for involving remote participants but, on the other hand, increases the costs of the knowledge and update, perhaps significantly.

Third, Rittle's influential approach to design rationale, Issue-Based Information Systems (IBIS), appeals to the idea that systematic argumentation will enable teams to manage the complexity of an unstructured design problem. As such, when performing under the rules of IBIS, a team generates a network of linked information units, each labeled according to its rhetorical purpose. In turn, the network's function is to be a:

> *documentation and reporting system which permits fast and reliable information on the state of discourse at any time* [12, p. 4].

In this aim, we see an optimistic focus on the low cost of knowledge. Experience using IBIS, and similar systems, however, shows that the cost of update is very significant. In fact, it is often so high that the use of such systems becomes impractical [10].

Fourth, empirical studies of open source software projects [6,9,19] have described the importance of relatively simple tools and usage policies, concerning such matters as how to report bugs, how to version code, how to report code changes, and so on. In fact, the community-based model of knowledge creation [13] proposes that code versioning (e.g., stable and experimental versions) together with a discussion space (e.g., a listserv) enables a social structure to develop (e.g., Project Leader, Maintainers, Developers, and Bug Reporters). The resulting sociotechnical system enables the community to enjoy its cumulative innovations by using stable releases while simultaneously allowing it to explore, evaluate, and learn using experimental releases of the code.

In this example, we can readily recognize separate spaces for *action* and *reflection* and deictic references between the two spaces [2]. Somehow, it seems reasonable to assume, the assembly and use of the information systems that underlie open source development strike a good balance between the costs of knowledge and update. The use of mundane tools and near invisible infrastructure is striking.

In summary, these wide ranging examples illustrate an important kind of meta-design – the design of information handling systems that support the capture, organization, and use of information on which design work depends. The *cost of knowledge and update* are concepts for thinking about this kind of meta-design.

## ENABLING "USER" PARTICIPATION IN DESIGN

Many approaches are currently emerging for inviting users to participate in design and development through various roles, such as the Monitored User, the Bug Reporter, the Remote Usability Participant, the Conceptual Innovator, and the Co-developer. Of course, to enact these roles and to take advantage of the information generated by them design information systems are needed.

*Bug trackers*, as one example, can be used to contribute structured feedback on errors [5], although pre-established labels for classifying errors, say operating system bugs, can make it difficult to submit and resolve other kinds of "errors," such as usability bug reports [17]. As a second example, Beta releases can use *discussion forums* to promote the formation of such roles as "lead users," "early adopters" and "innovators" [18, 15]. Getting into a Beta release, can garner social capital, which is then compounded when Beta participants write about their experiences in public forums and link to other Beta participants and technology commentators and culture shapers. Designers, in turn, can monitor these conversations at the periphery and develop an understanding for the users' perspectives, glean new ideas, or clarify design intuitions. Or, they may intervene directly in the forum and prompt users to talk about particular topics or share rationale on the system's evolving design.

Nevertheless, while these approaches provide the means for involving users, some significant questions arise about ends. How should the *development process* and the *artifact* under development be structured to accommodate *user-input* that is diverse and continuous? How should these three elements be interrelated? How, in short, does the network and such emerging technologies as those cited above expand the possibilities for involving users in a design and development environment and how does one select from all the possibilities? Finally, in turn, what new organizational capacities and individual design competencies are required?

Commercial slogans such as the *Participation Age* [16] and *Architecture of Participation* [14] give such questions particular urgency. The future possibility, in short, is: More people, of various roles, will participate with varying degrees of directness and influence – and most often remotely – in the development of information systems. From a position of scholarship, von Hippel [19] labels the trend *democratizing innovation* and provides evidence from a variety of domains that users are often the first to identify new needs and invent significant improvements. He argues that for commercial advantage, if not for social or ethical reasons, firms will need to structure product development to take full advantage of users' creativity and their situated adaptations of systems – but, how? As the call for this workshop asks – What are the sociotechnical conditions that lead to innovative and productive communities?

A noteworthy case for study is del.icio.us, a breakthrough application for social bookmarking. The development process for this socially oriented information management application is characteristic of open source software development in some respects and of proprietary, packaged software development in others. For example, while people are denied access to the del.icio.us source code and are prevented from running their own versions, they can use a public API to build their own applications that use the data held by del.icio.us. Then, people can discuss their innovations on a del.icio.us listserv and on public blogs. In turn, the del.icio.us development team can learn from others' development efforts and written reflections. Or, the development team can use the discussion spaces to prompt people to talk about their work in productive ways, to elicit new ideas and to discuss them, and so on. The broad picture that emerges is an intricate social network of joint reflection and the diffusion of ideas and tangible innovations (e.g., code fragments, user interfaces, etc.)

del.icio.us, in summary, shows how design and development can be *servitized*, leading to a process that proceeds simultaneously and is entwined with the formation and nurturing of a community of "users", perhaps more accurately called "innovators" [15]. And so design information systems will increase the information intensity of design as they expand the footprint of design process by allowing many more people to participate. Meta-design of this kind demands new competencies from individuals and new capacities from organizations.

## THE INFORMATION PERSPECTIVE ON DESIGN

The position I would like to advance in this workshop is that a deeper understanding for design can be obtained by taking an *information perspective* on design activities. Under this perspective the major unit of analysis is the information transaction – the specific needs and tasks associated with capturing, storing, updating, linking, and accessing information. I use the term Design Informatics to refer to this perspective.

By analyzing the information needs of design and how design teams create capacities to satisfy these needs, we may begin to recognize the invariant, technologically-neutral requirements that emerge from any design methodology. In turn, we are then able to recognize the commonalities of otherwise different methodologies. An information focus, in short, offers a strategy for developing a unified view of design. I wish to defend this claim and better understand it through vigorous dialog at the workshop.

## REFERENCES

1. Archer, B. Systematic Methods for Designers. In Cross, N. (Ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 1984.
2. Arias, E., Eden, H., Fischer, G., Gorman, A. and Scharff, E. Transcending the individual human mind: Creating shared understanding through collaborative design. *ACM Trans. on CHI*, 7, 1 (2000), 84-113.
3. Beck, K. et al. Manifesto for agile software development. Retrieved 01/10/07 from http://agilemanifesto.org/
4. Card, S.K., Pirolli, P. and Mackinlay, J.D., The cost-of-knowledge characteristic function: display evaluation for direct-walk dynamic information visualizations. In *Proc. CHI '94*, ACM Press (1994), 238-244.
5. Crowston, K. and Scozzi, B., Coordination practices with FLOSS development teams: The bug fixing process In *Proc. of the First International Workshop on Computer Supported Activity Coordination* (2004).
6. Ducheneaut, N., Socialization in an open-source software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 14 (2005), 323-368.
7. Green, T.R.G. and Benyon, D.R. The skull beneath the skin: entity-relationship models of information artifacts. *Int. J. of Human-Computer Studies*, 44 (1996), 801-828.
8. Grudin, J. Evaluating opportunities for Design Capture. In Moran, T. and Carroll, J. (Eds.), *Design Rationale: Concepts, methods and techniques*, Erlbaum, Mahwah, NJ, (1996) 453 - 470.
9. Hendry, D. G. Public Participation in Proprietary Software Development through User Roles and Discourse. Submitted.
10. Isenmann, S. and Reuter, W.D., IBIS: A convincing concept…but a lousy instrument? In *Proc. DIS '97*, ACM Press (1997), 163-172.
11. Kelley, T. *The Art of Innovation*. Random House, New York, 2001.
12. Kunz, W. and Rittel, H.W.J. Issues as Elements of Information Systems Working Paper No. 131, Institute of Urban and Regional Development, University of California, Berkeley, California, (1970).
13. Lee, K. G. and Cole, R. E., From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. *Organization Science*, 14, 6, (2003), 633-649.

14. O'Reilly, T. *The Architecture of Participation*, Retrieved 01.10.07 from www.oreillynet.com/pub/3017, (2003).

15. Rogers, E. M. *Diffusion of Innovations* (3rd Edition). Free Press, New York, 1983.

16. Schwartz, J. *The Participation Age*, Retreived 01.10.07 from http://blogs.sun.com/roller/page/jonathan?entry=inevitability, (2005, April 04).

17. Twidale, M.B. and Nichols, D.M., Exploring usability discussions in open source development. In *Proc. HICSS'05*, IEEE (2005), 198.3.

19. von Hipple, E. *Democratizing Innovation*. MIT Press, Cambridge, MA, 2005.

20. Yamauchi, Y., Yokozawa, M., Shinohara, T. and Ishida, T., Collaboration with lean media: How open-source software succeeds. In *Proc. CSCW '00* (2000), 329-33.