THE DATABASE REVISITED:

beyond the container metaphor

by

David Week
University of Sydney
Pacific Architecture, Pvt. Ltd.
Glebe, Australia.

DAVID WEEK
is an architect with clients whose cultures — indigenous, regional, organisational — are in transition. His firm, Pacific Architecture works in Papua New Guinea, New Zealand, Western Samoa, India, Scotland, Aboriginal and corporate Australia. He uses the Internet extensively, and is researching hermeneutics at the University of Sydney.

ABSTRACT

The growth of international networks, and of international trade in general, has increased the opportunities for architects to work together over distance. In our practice at Pacific Architecture, we've been using first modems, and now the Internet, to connect co-workers at sites in Australia, Oregon, Scotland, and Papua New Guinea.

Design collaboration has been primarily through the e-mail exchange of text and drawings. We've also assessed other CMC tools. Products like Timbuktu and video-conferencing software allow for real-time collaboration, based on the metaphor of two (or more) people together at a table, able to see and hear each other, and to work together on the same document. Groupware make intragroup communication the basis for building a workgroup's knowledgebase.

On recent projects, we've begun using database software as the basis for collaborative design communication. We've taken as a model for data structure Christopher Alexander's 'pattern language' schema. Conversations about the design take the form of a collaborative construction of the language. Inputs into the database are constrained by the 'pattern' format. The CAD drawings run in parallel, as an 'expression' or 'instance' of the language. So far, CAD and database do not have an integrated interface.

This paper describes our experience in these projects. It also outlines a set of design criteria for an integrated CAD/database environment economically and incrementally achievable within the constraints of currently available software.

Formulating such criteria requires the reconceptualisation of notions of ‘database’. This paper looks at these notions through philosophical and linguistic work on metaphor.

In conclusion, the paper analyses the way in which we can use a reframed notion of database to create a useful collaborative communication environment, centred on the architectural drawing.

A succession of primary metaphors

In an earlier paper, I’ve suggested a series of transitions in the primary metaphors we have used—in our practice—for thinking about computers [Week, 1995].

At first, computers were machines, and the paradigm was automation. Then, we came to think of them as media. Their role as media was overtaken by their use as communication tools. And finally, now, we think of them as shared workspace.

None of these metaphor exists to the exclusion of the others. Rather, over time, one comes to the fore, the others recede.

Each primary metaphor has a different ‘killer app’ which dominate our conception of the usefulness of the machine. For computers as machines, it was the programming languages that we used to control them. Then word processors, spreadsheets, and graphics programs led us to understand computers as media. For communications, e-mail and its associated compression, encoding, and terminal applications became central. Now, as we move from individual computing to group workspace, the essential application becomes groupware, and the essential groupware emerges as the database.

In summary:

metaphonr	‘killer app’
machines	programming languages
media	word Processors, spreadsheets, graphic programs (including CAD and DTP)
communication tools	email and associated applications
workspace	groupware - specialized databases

Most groupware, it seems, is aimed at the high-end of the corporate market. Programs such as Lotus Notes and DCA's OpenMind are expensive both in initial dollar cost and in programming time cost. We work in a Macintosh environment. Neither OpenMind nor Notes strongly support that environment.

For these reasons, we've started building our own groupware in Claris' FileMaker Pro. FileMaker is a flat file database [1]. It has a number of features, however, that allow it to perform in a close-to-relational way. Furthermore, it has the unique distinction among databases in that 85% of FileMaker databases are written by the end-user. As we will see below, this has proved to be important.

THE CONTAINER METAPHOR FOR THE DATABASE

Metaphor pervades our language, and—through our language—the way we see, think, and act. [2]

Mark Johnson, in The Body in the Mind, argues that metaphors can be categorised in terms of image-schemata [Johnson 1987]. These schemata, he argues further, arise out of our basic bodily experience.

There is an element of foundationalism in this argument…the implication that these schemata are 'given' to us, by a pre-existing reality—our bodies. I think its useful to accept Johnson's observation of repetitive patterns across metaphors, without taking aboard his attribution of metaphysical significance to these patterns.

One of the most pervasive of Johnson's schemata is what he calls the *container* schema. We deploy a container schema when we talk about something as though its a container, holding something else, being filled and emptied, bounding its contents from the greater world around it. We deploy a container schemata when we think of ourselves as filled with anger; when we think of the content of a sentence; when we think set-theoretically.

We talk about databases as containers that 'hold' data. A typical database consists of records. Each records consists of a number of fields. The fields can be empty, or be filled with data.

This image of a database has all the romance and charm of a filing system. There are metaphoric and historical roots to this similarity. A database can be conceived as a file drawer; the records individual files; and the fields like blanks in the forms which—when filled in—constitute the file.

It's for this reason, I think, that while we as architects were swept away by the graphical charm of the processed word, the numerical flexibility of spreadsheets and CAD, we left the database behind. The other applications were not only more graphical in their use, but they allowed greater flexibility in design. What could an architect love more than the opportunity to keep designing—whether in word, number, or line—until the last possible minute?

Shifting beyond the container

The concept of a database as filing system, as container for data, failed at the outset to exercise an appeal to our way of working. Since then, however, we in our practice have come to view the databases as central to our work as architects—even more so than CAD. The path by which the database was brought to our attention was as follows: 1 Like other firms, we maintain a database of clients, suppliers, staff, friends, and other contacts. This started life as a single-user, freeform database: Cassidy & Greene's Quickdex.

2 The level of computerisation of the firm increased, and the release of Apple's System 7 led to a networking capability built into the operating system. We started to experience the limits of Quickdex. The success of the database led it to grow. As the main conduit for contact with clients and others outside the firm, I maintained the database personally. Other people in the firm needed access to it. I was forced to be pass out contact data as well. This became onerous.

3 An initial workaround: copies of the database were periodically published—copies distributed across the net. Thus, others in the office could look up their own contact data. But as holder of the 'master' copy, I was still left responsible for all updating. We clearly need a shared database—a shared 'container' from which users could both add to and retrieve from. This led us to FileMaker Pro, which was both inexpensive and multi-user. We quickly established a shared FileMaker Pro contacts database.

The critical advantage that FileMaker had over Quickdex was the fact that it was *shared*.

A little after this move to FileMaker, we were presented with a project to write a pattern language for the development plan of a tertiary training institution . (I'll call this Tertiary Pattern Language the 'TPL' for short. For definitions of a pattern language, see Alexander 1965; Alexander et al, 1979; Alexander & Poyner, 1971.) One of the reasons that we were engaged to do this plan as a pattern language was that the client institution was responsible for dozens of campuses, and wanted to investigate methodologies that would allow them to share information across the plans for various campuses.

A pattern language is well-suited to this task—even designed for it. Normally, we would have used Microsoft Word or PageMaker in which to write the language. However, in this case, we saw that both:

- the exact printed format was critical, [3] and
- we couldn't fix that format, until after we'd written a substantial amount of the content

In Word or PageMaker, last minute changes in a large number of repetitive layout formats would not be easy. We realised that in FileMaker, this would be simple, since databases are designed to give a variety of different views into a common data space. *Whereas we originally conceived a database as a fixed structure, into which we could insert variable data, we now saw it as a flexible way of displaying a relatively fixed data space*. We altered the format of the database almost continuously, as the language evolved.

Furthermore, the use of FileMaker to write the language had the additional advantage—since the database was shared—that different people could work on different parts of the file at the same time: simultaneous co-authoring. From this point of view, the database is a medium—like CAD, word processing, and spreadsheet. It is different, however, in being commonly conceptualised and implemented as a shared medium—the others rarely so.

It's easy to understand that a FileMaker database can operate as a shared workspace, when used and conceived of in this way. To do so is not yet to go beyond the container metaphor for the database. As both Albert Einstein and Max Black have pointed out independently, we often conceive of space (and workspace) as a boundless container, in which objects sit (or work takes place.) [Einstein, 1921; Black, 1990]

But in the case of the TPL, the database was the work. In other words, it was not the 'content' of the database that was the finished product of our groups work, but the whole database itself—structure, content, forms.

In this understanding, 'data' and 'database' become conflated. *The work is the workspace*.

This conflation of data and data structure is possible only because FileMaker allows non-specialist users to make rapid changes in the database structure, and in associated forms, so that the database becomes as malleable as the data.

Dynamics of this progression

Before take the next step, it's worthwhile analysing this progression from certain theoretical perspectives.

Our practical encounter with the concept and technology of the database has many of the elements of a Kuhnian paradigm shift. It may seem that 'paradigm' is a very big notion through which to understand this very minor shift in thinking, but in many ways the terms 'paradigm' and 'metaphor' are interchangeable. Metaphor has no characteristic scale. '…metaphors are kind of microtexts, as texts are kinds of macrometaphors.' [Weinsheimer, 1991]

We begun by an unproblematically conceiving of the database according to the common container paradigm. That use proved successful, so we extended it. This extension took us into realms where breakdown begin to appear. The breakdown became increasingly untenable, and was resolved by reframing the concept of database, by first seeing the database as a workspace, and then as the work (data) itself.

This Kuhnian description of the dynamic is complemented by Hans-George Gadamer's notion of play. Per Gadamer, there is a to-and-fro play in dialogue, and it is through this play—not by design, not by teleology—that one metaphor is overturned in favour of another [Gadamer, 1975]. In this case, the participants in the to-and-fro were ourselves and the technology.

Finally, this dynamic can also be understood as a form of Derridean deconstruction. The metaphor of a database as a container holds within it a dichotomy. The database is split into the data structure (the container) and the data it holds. Normally, the data is privileged over the data structure. The data is of value, the active content of the database. The datastructure is passive recipient of the data.

In the process of re-interpreting the database as form for use in creating a document, we inverted the traditional privileging, making the data structure the active, valuable aspect of the database, through which we could successfully manipulate the publication of the data. Having inverted the opposition, it lost its hold over our imagination, and we lost the distinction completely, seeing data as inseparable—even indistinguishable—from data structure.

The key features of FileMaker Pro that made this transition possible—even drove it—were:

- the fact that is was multi-user, and allowed previously private data to be shared.
- that users can almost as easily manipulate the shape of the 'container'—the database structure—as they do the 'content'—the data itself.

Extension to drawing

Concurrent with this expansion of the uses of a database in our office, we were involved in a number of collaborative design exercises, in conjunction with a colleague in the UK. The first of these was a competition (we didn't win); the second a real project. This collaboration was mediated by e-mail.

After a rather hiccuppy starts, we quickly developed an effective working style which now stands as follows.

1. Chris Davenport (in Scotland) controls the drawings. He sends us a current version, in the form of a CAD document.
2. We (David Week and Stephen Loo, in Sydney) work over the version. We feed back information in the form of argument, and interpretation in terms of the drawings. This is done within the CAD document. We add a layer over a particular drawing or series of drawings, on which we put:
    - scanned sketches;
    - brief arguments, hypotheses, critiques, proposed principles;
    - roughly reworked sections of the drawing in question.
3. Chris Davenport then compiles these into a revised version, which is again sent to us.

The whole process may begin at either end.

Two salient features of this process:

- the collaborators assume more or less equal standing in the debate.

There is a symmetrical power relationship. Although the shifting power of rhetoric, experience, control of the medium, and so forth are always at play, this is not overlaid by a formal authority structure.

- the way in which the database is used is—in contrast—asymmetric. Chris Davenport accesses the formal drawing representation. We use primarily the informal drawings, and text annotations [4].

The whole process is asynchronous. This asynchronicity of the process is a major asset. Difference in timezone make synchronous processes difficult. In general, synchronous processes such as telephone calls and meetings are known and notorious timewasters, and need to be strictly limited and controlled [5]. Asynchronous processes (such as e-mail) allow each person to choose the timing of, and the amount of time spent in, their participation.

There are however major problems with this collaboration process:

1 *Modes*

The whole process follows a metaphor of correspondence. We mail messages back and forth. The whole process of composing mailings—writing, scanning, assembling—and compressing, encoding, and mailing them is too non-transparent, and time-consuming. It constitutes a separate activity—an interruption—from the fluid and productive thinking, sketching and dialogue that precedes it. This lack of transparency adds perhaps 50% overhead to the process—far too much.

Another way of saying this is that the medium is modal—either you are working, or you are composing/sending—the two modalities are separate. In this respect, it is like early word processors in which you had to switch between separate modes to write, or to edit, a distinction which has today disappeared.

We need that work/mail modality to similarly disappear in our group workspace.

2 Loss of history. The correspondence metaphor also leads us to work inefficiently from time to time, because we do not have ready access to the past. The document we typically build up through e-mail exchange is structured as one long string of correspondence. We forget why we did things; we violate decisions we made earlier; we re-cover old ground—all because we do not have ready and appropriate access to the history of our dialogue.

This is a problem that could be clearly resolved by an appropriate structure [6].

Interpreting the earlier lesson

The TPL had neither of these problems.

Working on the database was the same as communicating to a future reader or collaborator. One simply wrote. That was it. No modality.

The idea of getting rid of modes goes back to the early days of Xerox PARC, and was formulated by Alan Kay. It was incorporated into the Xerox Star, then the Apple Lisa and Macintosh, and today much software is modeless. The computer journalist Stephen Levy describes the significance of the absence—or at least the apparent absence—of modes, like this:

> 'As Kay later wrote, "This interaction [bringing windows forward by clicking on them] was modeless in a special sense of the word. The active window constituted a mode, to be sure—one window might hold a painting kit, another might hold text—but one could get to the next window to do something in without any special termination. This is what modeless came to mean for me—the user could always get to the next thing desired without any backing out."
>
> 'In other words, as far as the computer user was concerned, it was in a mode—but the user felt the freedom of modelessness. Switching windows was natural. If you view things a certain way, we're always switching modes, in everything we do…But in non-computer life, we don't have to stop and recite a secret word every time we change activities—look down from the television to the newspaper, watch the baseball and swing the bat—we just do it, and thus don't worry about what mode we're in.' [7]

Our problem with e-mail can be understood as a matter of modes.

In the TPL, there is also no loss of history, because the writing does not form one long time-ordered string, but is positioned in the database in its final place, where it is accessible to anyone else working in that area.

Of course, as text is edited or deleted, it can be lost. But that's a simple fix—a matter of pushing old versions of the content of a field down in a stack associated with that field, where they are immediate and available. This is far different from trying to find them in

a long, freeform string of correspondence, where even text search algorithms may miss the appropriate references.

Our international collaboration is more complicated than TPL in two ways:

1 it connects the participants by e-mail instead of on a LAN;
2 it makes the object of the work a drawing, rather than a database.

What do we need to do in order to bring the ease-of-use of the TPL to work centred on an architectural drawing, instead of a text document?

A COLLABORATIVE WEB SITE

In order to help resolve the first complication, we are replacing e-mail exchange with a collaborative Web site. The site will be private.[8]

This move away from e-mail exchange is critical. Although it might be possible to write a database with all the desired workspace features, and ship it back and forth using automated mail software, this would become increasingly resource-expensive as the database grew, and would never be extendible to the participation of more than two sites.

This Web site is being constructed as part of a funded research program at the University of Sydney Department of Architectural and Design Science. I am an associate investigator in this program. To start, the site software is Hypermail. Hypermail displays all mail sent to the site in a flexible list, and interprets HTML text sent to the site.

The strategy behind the site is that our firm will use the site for our collaborative work session, and in conjunction with my research colleagues, progressively adapt the software to get the features we desire. This is an iterative, interactive approach.

In its current incarnation, Hypermail is not adequate. Going from reading, to writing involves quite an onerous transition. Current support for drawings is also far from transparent. However, it does provide us with the base upon which to build.

DRAWING AS DATABASE

What shall we build at this site?

Compiling the various criteria and lessons from our earlier experience:

1 Modelessness, transparency, ‘readiness-to-hand.’

The site should be modeless. Certainly, there should be no onerous switching of modes between reading the database, and writing to it.

HTML allows for ‘forms’ input, in which input typed directly into a space on the Web page is processed by the site. You can witness this at work at WWW search engine sites (such as Lycos), and others Web pages using forms input for registration, order processing, etc.

2 Conflation of data and data structure

In working on TPL, it became the case that ‘the workspace is the work.’ Translated simply into the terms of an architectural drawing…‘the database is the drawing’ and ‘the drawing is the database’.[9]

In terms of interface, this means two things:

- the data structure should be as easily manipulated as the data
- the drawing—perceptually—fills the frame, so that the drawing is not ‘in’ the database, but rather other texts and data happen ‘in’ and ‘on’ the drawings—just as they do on a physical drawing.

3 Conflation of drawing and text

In TPL, the pattern format allowed for an useful structuring of an architectural conversation. In moving to collaborative authoring of a drawing, it’s important not to lose the advantage of the pattern format. But really, a drawing can be understood as text and graphics put together on a page. So too can a pattern. Both are cross-referenced to others of their own kind, and to each other.

Without denying the difference between a pattern and a drawing, conflating them in this way (deconstructing the distinction) has the advantage of allowing the creation of a very simple and isotropic workspace.

4 An accessible history

The history of the drawing should be readily accessible, in a number of ways:

- Localised input

Comments on a drawing should be situated so that they are close to the object to which they refer, so that a person looking at that part (or aspect) of a drawing can be easily led to pieces of its history.

- A historical stack
  Each drawing, and each pattern, in addition to its current version, should have an accessible stack of earlier versions.
- Flagged additions
  With e-mail exchange, we always know what mail has and hasn't been read. With a shared WWW site, recent changes in the database may not be so apparent, and will need to flagged somehow.

An imagined site

With these criteria in mind, I imagine a work session like this:

I open Netscape, and go to my firm's site.

There I find a list of current projects, some of them flagged to indicate that there have been additions to the discussion on these projects.

I choose one. It takes me to a drawing set. The drawings form the spine and centre of the architectural discussion. Only one participant in the project controls the drawings. All other participants contribute graphical or textual commentary.

The drawing set consists of a number of sheets. Each is a separate WWW page. The sheet is displayed as A3, at the same scale as if it were printed. I can use an A3 screen, or otherwise scroll around the sheet, to view it. To me, as I experience it through this interface, the drawing is not 'in' a database. It 'is' the database.

At one corner, a list of hypertext links will take me to other sheets in the drawing set, or out of the set altogether.

The list of links might look like this:

PROJECT XXXX

- site
- level 1 plan
- level 2 plan
- elevations
- model
- patterns

  EXIT to project list

I'm able to add links — create new sheets in the drawings set. Let's say I'm looking at a particular sheet: the elevations. Another list of links allows me to view the history of the drawing.

The list might look like this:

- ELEVATIONS v3
- commentary XX
- ELEVATIONS v2
- commentary YY
- commentary XX
- ELEVATIONS v1

  EXIT to drawing list

If I click on any elevation, I see just that sheet. But if I click on a commentary, I have the choice of seeing that commentary either alone, or with the underlying elevation visible beneath it. In other words, I can see the commentary on a drawing either alone or in context.

Let's say that I go to an existing commentary. I see there—overlaid on the subject of the comments—text and scanned freehand sketches. The text can be typed and located in Netscape (using the forms facility). Scanned sketches have to be uploaded first, and then referenced in some simple way.

Each comment in the commentary is linked to one or more Patterns. Clicking the link takes me to the Pattern, which is itself another commentary sheet (unrelated to any one drawing sheet). The pattern sheets are collections of comments on drawings, plus comments local to the pattern sheet itself, plus graphics of both varieties. In this way, each pattern is also linked back to the drawings to which it relates.

Summary of data structure and functionality

What is the structure of this database?

Each record is a sheet. On each sheet, I am able to place text or graphic blocks, and locate them spatially. Within those blocks, certain pieces of text are links to other sheets.

Some sheets are labelled as Patterns, some as Drawings, others as Commentary. Beneath the label, however, they are structurally the same.

The user interface allows the user to place text and scanned images on these sheets, and to delineate some text as links to other sheets. Periodically, the controller of the drawings reads the new commentaries, uses them the alter the drawing, and uploads a new version (possibly with a commentary layer, explaining his or her interpretation, and reasoning behind the version.)

* * *

This design for a collaborative work environment seems to comply with all the criteria elicited from our earlier experience. The resulting environment is a shared database. It is not however, the simple empty container implied by the classical concept of a database record or field, or by the notions of ‘space’ implicit in ‘workspace’. Rather, the datastructure is a rich, malleable material, whose final form is as much a part of the ‘content’ of the database as text or graphics.

Thinking somewhat self-consciously about our experience with previous database designs, coupled with a theoretically understanding of concepts as metaphors, and the forms of their transformation, has allowed us to design a new database, in a structured way.

Postscript: About patterns

One of my colleagues in this group of papers raised concerns about patterns as a form of structuring architectural dialogue. First, the endless conversations that seemed in his experience to go into formulating a pattern. Second, the further problem which seemed to remain unresolved once a pattern was formulated, of fitting it into a particular context.

These problems seem to me to flow from a particular vision of patterns as universal laws, which then get applied to concrete situations. If you are attempting to formulate a general principle of architecture, the scope is opened to talk for a very long time indeed.

Furthermore, when a pattern is formulated in this way, it is unrelated to any particular situation, and therefore the work of drawing out that relation has yet to be done.

All of this takes place within a frame in which designing is understood as a scientific activity.

The frame in which I am operating sees design as a conversational activity, in which the divergent horizons brought to the situation by the various actors (animate and inanimate)are fused through the play of dialogue.[10] Patterns are merely a helpful ways of structuring the records of the dialogue—nothing either more or less profound.

Patterns considered this way do not go on forever, because they are not a search for an unreachable ideal. Furthermore, they are always come out of a conversation about a concrete aspect of the design, so that the connection to the particular place does not have to be made later—it's already made. And—as we know—conversations about concrete questions tend to settle down sooner that conversations about abstract generalities.

ENDNOTES

1 Claris has announced that the next version (3.0) will be relational.

2 This pervasiveness of metaphor is well described in [Lakoff & Johnson, 1980]. As a theme, it runs through recent European philosophy. A detailed analysis of a particular case is given by [Reddy, 1979]. For an analysis with particular reference to design, see [Schön, 1979].

3 Our clients confirmed that most development plans, once written, were consigned to a shelf never to be read again. Only the physical drawing continued to be used as a method for locating buildings. It seemed important to make the patterns very short, punchy, graphically clear. In addition, the desire to re-use patterns across master plans implied a highly standardised format.

4 A formal architectural drawing is a complex object. Leaving it in the hands of one participant minimises the chances of inconsistences appearing in the drawing set, and is very efficient, as the drawing is always manipulated by someone understands its construction. This asymmetry builds in some biases—after all, Chris does end up controlling the drawings. But this is countered by the fact

that David and Steve are assume the position of critics and instructors, and are freed of the onus of making finely detailed, consistent drawings.

5 See almost any book on time management.

6 Horst Rittel's IBIS comes to mind. Rittel's 'issues' and 'positions' however, are based on argumentative categories, whereas patterns are based on spatial configurations, and are therefore at least in principle more easily tied to drawings. Thus, for instance, the issue of 'security' is not tied to any particular spatial attribute of the design, where as the pattern 'hierarchy of zones' is.

7 [Levy, 1994] p61-62. See also [Cringely, 1992] p193: 'Modes were evil. At PARC you were either modeless or impure…' Levy's description of modelessness is strikingly similar to the Heidegerrian notion of 'readiness-to-hand'.

8 A private Web site is not really a contradition in terms: simply a site to which no other sites are linked, and therefore to which only those privy to the address can go.

9 In a trivial sense, all CAD drawings are databases. They're also (quite often) computer programs. These senses of this conflation are trivial, because the issue is not how the drawings present to a programmer, but how they present to the architectural user.

10 See Bruno Latour for analyses of scientific experimentation as conversations between animate and inanimate witness and actors. See for example [Latour, 1991; Latour, 1989]. For a summary of horizons, fusion of horizons, and the play of dialogue, see [Bernstein, 1983].

BIBLIOGRAPHY

Alexander, Christopher & Poyner, Barry, “The atoms of environmental structure,” Emerging methods inn Environmental Design and Planning. edited by Gary Moore (Cambridge: MIT Press, 1971)

Alexander, Christopher, et al., A Pattern Language, (Oxford: Oxford University Press, 1979)

Alexander, Christopher, “A city is not a tree,” Architectural Forum (April-May 1965): 58-62

Bernstein, R. J., Beyond Objectivism and Relativism: Science, Hermeneutics, and Praxis (Oxford: Basil Blackwell 1983)

Black, Max, “On demystifying space,” Perplexities (Ithaca: Cornell University Press, 1990): 165-173 Cringely, Robert X., Accidental Empires (London: Penguin, 1992): 193

Einstein, Albert, “Geometry and experience,” Ideas and opinions: Albert Einstein, edited by Selig, C. (New York: Crown Publishers, 1921): 232-246 Gadamer, H.-G., Truth and Method (London: Sheed and Ward, 1975)

Johnson, M. L., The Body in the Mind: the Bodily Basis of Meaning, Imagination and Reason (Chicago: Chicago University Press, 1987) Lakoff, George & Johnson, Mark, Metaphors We Live By (Chicago: University of Chicago Press, 1980)

Latour, Bruno, “Clothing the naked truth,” Dismantling Truth, edited by . In Hilary Lawson, H. and Appignanesi, L. (London: Weidenfeld and Nicolson, 1989): 101-126

Latour, Bruno, We Were Never Modern (New York: Harvester Wheatsheaf, 1991) Levy, Stephen, Insanely Great (New York: Viking, 1994): 61-62

Reddy, M. J., “The conduit metaphor—a case of frame conflict in our language about kanguage,” Metaphor and Thought, edited by A. Ortony, A. (Cambridge: Cambridge University Press, 1979): 284-324

Schön, D. “Generative metaphor: a perspective on problem-setting in social policy,” Metaphor and Thought, edited by Ortony, A. (Cambridge: Cambridge University Press, 1979): 254-282

Week, David, “The Message is the Medium,”ACADIA Quarterly (Vol 14, No 1, 1995): 21-27

Weinsheimer, J., Philosophical Hermeneutics and Literary Theory (New Haven & London: Yale University Press, 1991).