# The TAC Paradigm: Unified Conceptual Framework to Represent Tangible User Interfaces

Eduardo H. Calvillo-Gámez Nancy Leland Orit Shaer Robert J.K. Jacob
Department of Computer Science
Tufts University
Medford, MA, USA
+1-617-627-3217
{calvillo, nleland, oshaer, jacob}@cs.tufts.edu

## ABSTRACT
This paper introduces a new paradigm for describing Tangible User Interfaces (TUI). The paradigm presented here encompasses existing TUI classifications and proposes a unified conceptual framework with which all TUIs can be understood. In order to show that the new paradigm holds and can be generalized we analyzed several existing TUIs using the proposed paradigm.

**KEYWORDS:** Tangible User Interface (TUI), TAC, Pyfo

## INTRODUCTION
Since the introduction of Tangible User Interfaces (TUI) by Ishii and Ullmer [5] in 1997, many systems implementing TUIs have been developed. Most of these systems focus on solving specific tasks. Little effort has been aimed at defining the conceptual framework behind TUIs. In this paper we present a new paradigm for understanding TUIs, which we call the TAC paradigm. The TAC paradigm encompasses the existing classifications and provides designers a unified conceptual framework for describing TUIs, which could provide a basis for creating a toolkit for designing TUIs.

To date, Holmquist [4] and Ullmer [15, 17] realised the most comprehensive work on defining a framework for TUIs. Since Ullmer included Holmquist's input, throughout this paper we will refer to the former. Ullmer proposed three classifications for TUIs: Interactive Surfaces, Constructive Assemblies and Token+Constraints.

An interactive surface allows the user to interact with tangible devices on an augmented planar surface. Two subsets of interactive surfaces are Interactive Workbenches and Interactive Walls. Interactive surfaces have been most compelling in spatial or geometric application domains such as urban planning [18], geographical space [16] and assembly line planning [11]. There are also several TUIs that represent an abstract application domain, such as the Senseboard [6] and the Designers Outpost [7].

Constructive assemblies were inspired by the Lego$^{TM}$ style interface. This type of interface is constructed by connecting "blocks"; each block may have electronic data associated with it. This approach is often employed to express models of the physical world [1] or to describe abstract structural logic such as programming languages [13, 9].

In the Token+Constraints classification each physical object (or token) is constrained by other physical object(s) (or constraint(s)). The token+constraint approach centers on hierarchical relationship between two kinds of physical elements: tokens and constraint [15]. Tokens can be attached or detached from physically compatible constraints. Manipulation of tokens takes place within the physical bounds of constraints, and consists of association and manipulation phases. Tokens may be moved from one constraint to another, thus altering the digital operations. Constraints may contain multiple tokens, and tokens and constraint relationship can be nested. Examples of existing TUIs that fit this approach are LogJam [3] a video logging interface where physical blocks represent categories of video annotations and ToonTown [12] audio conferencing where physical objects represent users. Both of these systems use a multi-tier rack structure as a constraint. Tangible query systems use physical objects to represent parameters and query racks or query pads as constraints [15]. WebStickers use physical objects to represent web pages [8].

Ullmer recognized that his classifications do not cover all existing TUIs; some existing TUIs do not fit into any of his classifications, and others overlap classifications. Commenting on this he says that the three classifications "were intended neither as a taxonomy nor as an exhaustive description of the TUI design space" [15]. His intent was that the classifications would "provide a starting point for considering TUIs not as isolated systems but as related elements of a larger design space" [15].

Svanaes [14] suggested that it is time to look for new metaphors that accurately represent TUIs; to find the “widgets” for TUIs. Being able to discuss and understand TUIs within a unified framework is the first step forward towards finding the ”widgets” and developing toolkits for building and testing TUIs.

The remainder of this paper is organized as follows: First we describe the proposed TAC paradigm. Next, we analyze several representative TUIs using the new paradigm. Three of the TUIs we analyze represent Ullmer’s classifications. In addition, we analyze a TUI that falls outside Ullmer’s classifications and a TUI that overlaps classifications. Finally, the last section contains our conclusions and future work.

## THE TAC PARADIGM

The TAC Paradigm uses the Token+Constraint classification as its base then builds upon it. We expand the concept of constraining a physical object by association and manipulation; in the TAC paradigm any object may be constrained regardless of its shape or nature. Additionally, we allow TUIs to have either discrete or continuous interactions. One problem we faced was the lack of a standard language for describing TUIs, hence we introduce the following terminology. Next, we present the basic properties of every TUI.

### TAC Terminology

**Pyfo** A physical object.

**Constraint** A pyfo that limits the behavior of other pyfo(s).

**Token** A pyfo with which a user has tangible interaction, performs a task and is limited by constraints. The task performed has a direct impact on the TUI’s application.

**TAC** A token and its constraints.

**Variable** Digital information associated with TACs.

### Properties

The TAC paradigm has the following five properties:

**First,** *A pyfo must be associated with a constraint and a variable in order to be considered a token.* For example, in the case of a ball inside a box, the ball’s possible range of motion is constrained by the box. However, the ball itself is not a token until is associated with some variable that changes the status of the application.

**Second,** *Each pyfo may be defined as a token, a constraint or both.* In the situation where two pyfos are both, a token is a constraint for its constraint, and the constraint is also a token we consider these two different TACs, even though they use the same pyfos. In this case, two independent sets of relationships are maintained. Consider the example of a ball in a box. The ball is constrained by the box in that its range of motion is limited to the dimensions of the box. However, when we shrink the box, the box is a token and the ball becomes the constraint because the box cannot shrink smaller than size of the ball. In this situation we have two separate TACs, ball in a box, and a box around a ball.

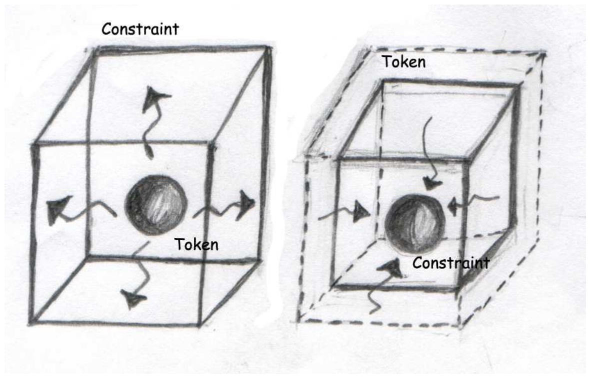

Figure 1: First and Second Properties. Here each pyfo acts both as a token and constraint. In the picture to the left, the box is constraining the bouncing of the ball. In the picture to the right, the ball is constraining the shriking of the box. The same two pyfos are member of two different TACs.

**Third,** *A token must keep a list of its constraints, but a constraint doesn’t keep a list of its tokens.* Using the ball in a box example, other than limiting the range of the ball’s motion, the box has no influence on the status of the TUI or the application. For the box, it’s no different if there is one ball or five balls bouncing inside. However, for the ball, the number of balls inside and the dimensions of the box will matter. Additional balls will further hamper motion, and thus adding balls increasingly restricts the ball’s range of motion.

**Fourth,** *There is a variable associated with every TAC.* Physically interacting with a TAC causes the status of the application to be altered. The user observes the changes in the form of feedback from the application. The type of feedback received depends on what was specified by the designer of the application. For example, pressing a box with a ball in it could cause the box to shrink, or it could cause the box (and ball) to simply move locations. Additionally, there may be a variable “bounce” associated with the ball. In this case, the status of the application changes as the ball goes from sitting stationary inside the box to bouncing around inside the box.

**Fifth,** *Each TAC has either a discrete or continuous physical behavior.* Each behavior is the response to a discrete stimulus. For example, a pushbutton can only be pushed (discrete behavior), and is activated when the user presses the button. A knob can be turned (continuous behavior), but the continuous act of turning is initiated in response to the user grabbing the knob (discrete stimuli).

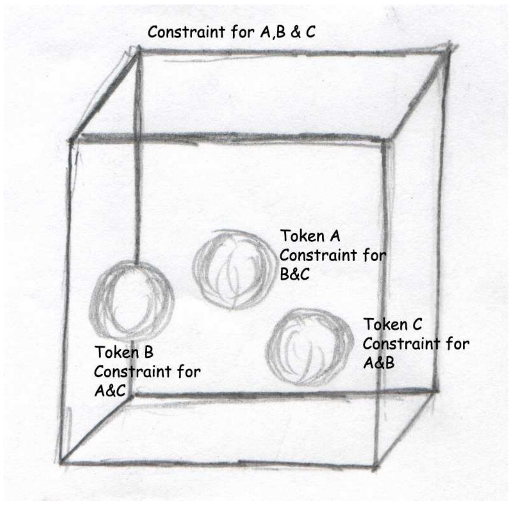

Figure 2: Each Pyfo can be a token, a constraint or both of them. In the above example, each ball is both a token and a constraint, and the box is only a constraint.

It is important to note that the nature of the TAC paradigm is to divide each TUI into a set of unique TACs. Figures 1 and 2 are a graphical representation from the above examples.

## ANALYZING TUIS WITH THE TAC PARADIGM

Having proposed a new paradigm, the next step is to determine whether the paradigm has the ability to describe existing TUIs.

In this section we look at several representative TUIs and describe them using the TAC paradigm. We selected existing TUIs that fall under each of Ullmer's classifications, Webstickers [8] as a Token+Constraint, The Designers Outpost [7] as an Interactive Surface and Computational Building Block [1] as Constructive Assemblies. In addition, we analyze one interface that does not clearly fit any of the existing classifications, ComTouch [2], and one interface that overlaps Ullmer's classifications, Illuminating Clay [10].

We analyze each TUI in terms of association and behavior. Association refers to both the tangible composition and the physical motion of each TAC. In it we describe the members of each TAC (the token, its constraints and its variable). The behavior analysis refers to the interaction associated with each TAC and the feedback produced by them. We summarize our analysis of the TUIs in tables 1 to 5.

### Webstickers

Webstickers is a system that provides users a way to tangibly access web pages. In Webstickers, a physical object is associated with a web page. The physical object can be anything from a Sticky-note to a baseball cap. The physical objects serves as a reminder of the web page link. For example, an ACM mug may be linked to the ACM web page. The system is implemented using a simple bar code reader.

Webstickers falls into Ullmer's Token+Constraint classification. Examining the system, one sees that the physical objects (either a Sticky-Notes or some other physical representation (referred to by the creators as PhyIcons)) behave as tokens. Sticky-notes, for example, can be placed anywhere, on a desk, on a book etc., and thus are associated with some constraint (whatever it is sticking to). Also, there is a manipulation interaction when moving the PhyIcon to the barcode reader.

In the TAC paradigm, the sticky-notes or PhyIcons are pyfos with two constraints. One constraint is the users' hand, because the user must grab the token in order to scan it into the computer. The barcode reader is also a constraint, because for a web page to be opened, the token must be scanned.

Two important characteristics are seen here: First, we consider the hand a passive constraint since using your hands is implicit to TUIs. Second, in this case, we consider the barcode reader a constraint and not just a technological limitation. The barcode reader limits the range of movement of a given token. A token must be within readable range of the barcode reader if a web page is to be opened.

Table 1 summarizes the websticker analysis using the TAC paradigm.

### The Designers Outpost

The Designers' Outpost is a tangible interface for collaborative web site design [7]. The system tracks sticky notes as users physically add, remove and move them around a board. Each sticky note represents a web page and may eventually be replaced by an electronic note. When the user taps on a sticky note an electronic menu is presented, allowing a user to replace the sticky note with an electronic one or to delete the note. A tool tray is attached to the board and contains an electronic pen, an eraser and a move tool. A link between notes is created using the electronic pen. Electronic notes can be moved with the move tool, and the eraser is used for erasing electronic ink.

The user may also use the electronic pen for free form drawing. Deleting or moving an electronic note is done with the eraser or the move tool. The Designer's Outpost is a classic example of ”interactive surfaces” [15].

In the TAC paradigm, the sticky note, the electronic note (enote) and the tools are identified as tokens. Each fit the definition of a token in that they are directly manipulated by the user, maintain a list of constraints, and are attached to a variable in the application.

An interesting aspect of this system is that the constraint list maintained by the tokens is dynamic and changes according to the state of the token. For example when the eraser is not active it is constrained by the tool tray but not by the board. When active, it is constrained by the board and the list of

| *TAC* | *Association* | | | *Behavior* | |
|---|---|---|---|---|---|
| | Token | Constraints | Variable | Action | Feedback |
| 1 | Sticky-Note / PhyIcon | Barcode Reader Hand (Passive) | Webpage | Scan | Webpage Displayed |

Table 1: Analyzing Webstickers using the TAC paradigm

| *TAC* | *Association* | | | *Behavior* | |
|---|---|---|---|---|---|
| | Token | Constraints | Variable | Action | Feedback |
| 1 | Note | Board<br>List of Notes | Webpage | Add | Add Note |
| | | | | Remove | Remove Note |
| | | | | Tap | Show Menu |
| 2 | Enote | Board<br>List of Notes | Webpage | Add | Add Enote |
| | | | | Erase | Erase note |
| 3 | Eraser | Board<br>List of Notes<br>Tool Tray | Webpage | Disconnect from Tray | Activate Eraser |
| | | | | Connect to Tray | Deactivate Eraser |
| | | | | Erase | Erase E-ink |
| 4 | Pen | Board | Link<br>Form | Disconect from Tray | Activate Pen |
| | | | | Connect to Tray | Deactivate Pen |
| | | | | Link Notes | Show link |
| | | | | Draw | Show Form |
| 5 | Move Tool | Board | Webpage | Disconnect from Tray | Activate Move Tool |
| | | List of Notes | | Connect to Tray | Deactivate Move Tool |
| | | Tool Tray | | Move Enote | Move E-note |
| 6 | Emenu | Note | Webpage | Close | Close Menu |
| | | | | Tap | Add or Release Note |

Table 2: Analyzing The Designers Outpost using the TAC paradigm

notes but not by the tool tray.

Table 2 is a summary of the The Designers' Outpost system using the TAC paradigm.

**Computational Building Blocks**

Computational Building Blocks is a system allowing users to construct a structure using Lego$^{TM}$ type building blocks, which will later be displayed graphically on a computer screen. Each of the blocks is encoded with information about its shape, color and texture. Each block also is aware of those blocks it considers "neighbors". A neighbor is any block physically connected to a given block. When a structure is assembled, and the power is turned on, each block identifies its neighbors. The neighbor information and the individual attributes of each block are sent as input to geometric computation functions.

The interesting aspect of this system is that blocks do not pass information directly to the computer, but instead pass the information to their neighbors, who in turn pass it to their neighbors etc. In the end, there is one block connected to the computer, and this block sends all the data it receives to a host computer. Computations are run on the data, and the output is a graphical interpretation of the physical structure, displayed on a computer screen.

Computational Building Blocks is an example of Ullmer's constructive assemblies category. The result being a graphical representation of the structure assembled. It is intended that the structure assembled represent a building of some sort, and the electronic data associated with each block is specifically intended to aid in understanding the structure as a model of a building.

According to the TAC paradigm, Computational Building Blocks is interesting because a block is a token, a constraint and a variable. A block is a constraint when it is under another block, and thus constraining the movement of the block above. In some cases, a single block may have two blocks constraining it (as in the case of corners on a building). A block is a token when it is added or removed. A block is a variable when it represents a digital block.

| *TAC* | *Association* | | | *Behavior* | |
|---|---|---|---|---|---|
| | Token | Constraints | Variable | Action | Feedback |
| 1 | Block | Block(s)<br><br>Table<br>Hand (Passive) | Neighbor Blocks<br><br>Computation Functions | Add<br><br>Remove | Physical Structure altered |

Table 3: Analyzing Computational Building Block using the TAC paradigm

| *TAC* | *Association* | | | *Behavior* | |
|---|---|---|---|---|---|
| | Token | Constraints | Variable | Action | Feedback |
| 1 | Cell Phone | Hand | Emotion | Vibrate | None |
| 2 | Hand | Cell Phone | Emotion | Press | None |

Table 4: Analyzing ComTouch using the TAC paradigm

Table 3 is a summary of Computational Building Blocks using the TAC paradigm.

### ComTouch

ComTouch is a TUI with the unique characteristic of giving tangible feedback uninitiated by the user. The system works like a normal cell phone, with the additional property of being able to ”express emotions” (using a half-duplex technology). When user A is talking to user B, he is able to supplement his verbal communication by squeezing the cell phone. When User A squeezes his cell phone, User B’s cell phone vibrates, the intensity of the vibration being indicative of the emotional intensity. The system sends different levels of vibrations/emotions (intense, less intense, etc), depending on how hard the transmitting phone is squeezed.

Comtouch is difficult to identify using Ullmer’s classifications. It’s not a constructive assembly or an enhanced surface. Neither is it comfortably placed within the Token+Constraint classification, since no clear constraints for the association and manipulation interactions can be identified. One could argue that the hand acts as a constraint. However, under Ullmer’s Token+Constraint classification, the hand is not considered a constraint but rather a tool used only to attach a token to a constraint(s).

Looking at our analysis more closely, in this system the hand is not considered a passive constraint, because the hand also receives output from the system. In the example above, when user A squeezes his cell phone, his hand is the token and it is constrained by the cell phone. The cell phone has a limit to how hard it may be squeezed. In the case of User B, his hand is the constraint; he receives a vibration that is constrained by his hand.

There is no system feedback to the user who squeezes the cell phone from the user receiving the vibration. If User A keeps squeezing and User B doesn’t respond, User A may conclude the system is broken. In this case, the feedback is independent from the TUI.

Table 4 summarizes ComTouch using the TAC paradigm.

### Illuminating Clay

Illuminating Clay is a TUI for landscape analysis. Users are able to alter the topography of a clay landscape model. Changes to the topography are captured by a laser, and the resulting depth image of the model acts as input to a group of landscape analysis functions. The results of these functions are displayed back on the table, both on the model itself, and additionally, results of several analysis functions may be displayed on the table next to the clay model. The user can change the functions displayed by selecting desired functions with a mouse. In addition, the user is able to select a particular area of the model with a mouse, and the cross section of that area will be displayed on the table. Any random object, such as a stapler, a role of tape etc., can be added to the model to help with analysis. The object for example, may represent a structure that casts shadows, and thus aid analysis of the amount of sunlight crops will receive if the structure is present.

Illuminating clay is a cross between Ullmer’s constructive assemblies and interactive surfaces. Here, interaction is taking place on the table surface, but yet, the topography or literally, how much clay is stacked up on top of each other has a direct impact on the analysis functions. Also, adding other physical items on top of the topography directly impacts the system. Finally, the TUI is intended to express the physical topography of a particular landscape.

Looking closer at our analysis, in Illuminating Clay, it is the laser which notes changes and feeds those changes to the analysis functions. The sculpting action does not directly

| *TAC* | *Association* | | | *Behavior* | |
|---|---|---|---|---|---|
| | Token | Constraints | Variable | Action | Feedback |
| 1 | Hand | Clay | Model Surface<br>Analysis Functions<br>Sections | Sculpt | Displays Updated |
| 2 | Hand | Mouse | Position<br>Model Surface<br>Analysis Function<br>Sectuins | Move | Mouse<br>position on<br>model changes |
| | | | | Click | Displays Updated |
| 3 | Random<br>Objects | Clay<br>Hand (Passive) | Model Surface<br>Analysis Functions<br>Sections | Add<br>Remove | Random Object<br>Added/Removed<br>to/from Clay Model<br>Displays Updated |

Table 5: Analyzing Illuminating Clay using the TAC paradigm

cause any function analysis. However, since the laser could be replaced with another technology, and since it is not directly part of the system - we have chosen not to include it in our analysis. In addition, while intuitively one would think that the act of adding or removing clay would be a special case. In reality, it is not. In fact, the actions squish and press both are a combination of remove and then add. For example pressing clay removes clay from one place and adds it to another. The results of all actions are identical; the changes in topography are input into the analysis functions and those results projected back onto the surface of the model and the table. Thus, all actions have been reduced to sculpt.

When adding or removing objects to/from the surface of the clay, we recognize two TACs: an object and the clay, and the object and the hand. The hand is, again, a passive constraint, which adds or removes objects to/from the clay model.

Table 5 summarizes Illuminating Clay system using the TAC paradigm.

## CONCLUSIONS AND FUTURE WORK

In this work we introduced a new paradigm, the TAC paradigm to describe TUIs. Having analyzed the five representative TUIs, we observe that the paradigm does indeed hold in these cases, which suggests that the TAC paradigm can be generalized for future TUIs. This paradigm may be derived into a methodology for designing TUIs in three phases: Description, Association and Definition. Description, defines the geometry and the physical properties of a pyfo. Association, combines pyfos into TACs, and specifies their actions. Definition, defines application variables attached to the TACs and specifies behavior.

In order to implement this design methodology as a toolkit, a high level description language needs to be created for describing the behavior of TACs and their effect on the application. From our analysis we learned that the TAC paradigm is object oriented by nature, and thus provides smooth transition from a conceptual framework to a conceptual architecture for TUI software. Each pyfo represents a class derived from an abstract pyfo class containing geometric and physical properties. When a pyfo is a token, it inherits from an abstract token class a list of constraints and the possible actions to which it responds.

All these concerns are left for future work. From creating a high description language, to specifying the behavior of TACs, to finding design patterns that can be used in TUIs software architecture, to developing toolkits, simulation and testing environments and applying TUIs to a wider variety of application domains.

## ACKNOWLEDGMENTS

We thank Lars Erik Holmquist and Brygg Ullmer for valuable comments on previous drafts of this document.

## REFERENCES

1. Anderson, D., Frankel, J., Marks, J., Agarwala, A., Beardsley, P., Hodgins, J., Leigh, D., Ryall, K., Sullivan, E., and Yedidia, J. "Tangible Interaction + Graphical Interpretation: A New Approach to 3D Modelling". In *International Conference on Computer Graphics and Interactive Techniques*, 2000.

2. Chang, A., O'Modhrain, S., Jacob, R.J.K., Gunther, E., and Ishii, H. "ComTouch: Design of a Vibrotactile Communication Device". In *Designing Interactive Systems Conference*, 2002.

3. Cohen, J., Withgott, M., and Piernot, P. "Logjam: A Tangible Multi-Person Interface for Video Logging.". In *Conference on Human Factors and Computing Systems*, pages 128–35, 1999.

4. Holmquist, L.E. *"Breaking the Screen Barrier"*. PhD thesis, Götebor University, Sweeden, May 2000.

5. Ishii, H., and Ullmer, B. "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms". In

*Conference on Human Factors and Computing Systems*, March 1997.

6. Jacob, R.J.K., Ishii, H., Pangaro, G., and Patten, J. "A Tangible Interface for Organizing Information Using a Grid". In *Conference on Human Factors and Computing Systems*, pages 339–46, 2002.

7. Klemmer, S.R., Newman, M.W., Farrell, R., Bilezikjian, M., and Landay, J.A. "The Designers' Outpost: A Tangible Interface for Collaborative Web Site Design". In *Symposium on User Interface Software and Technology*, pages 1–10, 2001.

8. Ljungstrand, P., Redström, J., and Holmquist, L.E. "WebStickers: Using Physical Tokens to Access, Manage and Share Bookmarks to the Web". In *Conference on Designing Augmented Reality Enviroments*, April 2000.

9. McNerney, T. "Tangible Programming Bricks: An Approach to Making Programming Accessible to Everyone". Master's thesis, Massachusetts Institute of Technology, 2000.

10. Pipper, B., Ratti, C., and Ishii, H. "Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis". In *Conference on Human Factors and Computing Systems*, 2002.

11. Schäfer, K., Brauer, V., and Bruns, W. "A New Approach to Human-Computer-Interaction Synchronous modelling in real and virtual spaces". In *DIS'97*, pages 335–44, 1997.

12. Singer, A., Hindus, D., Stifelman, L., and White, S. "Tangible Progress: Less is More in Somewire Audio Spaces.". In *Conference on Human Factors and Computing Systems*, pages 104–11, 1999.

13. Suzuki, H., and Kato, H. "AlgoBlock: a Tangible Programming Language, a Tool for Collaborative Learning". In *Proceedings of the Fourth European Logo Conference*, pages 297–303, 1993.

14. Svanaes, D., and Verplank, W. "In Search of Metaphors for Tangible User Interfaces". In *Conference on Designing Augmented Reality Enviroments*, 2000.

15. Ullmer, B. *"Tangible Interfaces for Manipulation Aggregates of Digital Information"*. PhD thesis, Massachusetts Institute of Technology, September 2002.

16. Ullmer, B., and Ishii, H. "The metaDESK: Models and Prototypes for Tangible User Interfaces". In *Symposium on User Interface Software and Technology*, pages 223–32, March 1997.

17. Ullmer, B., and Ishii H. "Emerging Frameworks for Tangible User Interfaces". *IBM Systems*, 39(3 and 4), 2000.

18. Underkoffler, U., and Ishii, H. "URP: A Luminous-Tangible Workbench for Urban Planning and Design". In *Conference on Human Factors and Computing Systems*, pages 386–93, 1999.