Pers Ubiquit Comput (2004) 8: 359–369
DOI 10.1007/s00779-004-0298-3

ORIGINAL ARTICLE

**Orit Shaer · Nancy Leland · Eduardo H. Calvillo-Gamez
Robert J. K. Jacob**

The TAC paradigm: specifying tangible user interfaces

Received: 1 November 2003 / Accepted: 19 April 2004 / Published online: 22 July 2004

Abstract This paper introduces a paradigm for describing and specifying tangible user interfaces (TUIs). The proposed Token and Constraints (TAC) paradigm captures the core components of TUIs while addressing many of the conceptual challenges unique to building these interfaces. The paradigm enables the description of a broad range of TUIs by providing a common set of constructs. Thus, the TAC paradigm lays the foundation for a high-level description language and a software toolkit for TUIs. We evaluate the proposed paradigm by testing its ability to specify a wide variety of existing TUIs.

Keywords Tangible user interface · Physical computing · Token and constraints · User interface management system · User interface description language · Software toolkit

1 Introduction

The last decade has seen a wave of new research aimed at fusing the physical and digital worlds. This work has led to the development of a collection of interfaces allowing users to take advantage of their spatial skills and to interact collaboratively with augmented physical objects in order to access and manipulate digital information. These interfaces are referred to as Tangible User Interfaces (TUIs) [1].

Interaction with TUIs draws on a user's existing skills of interaction with the real world, thereby, offering the promise of interfaces that are quicker to learn and easier to use. However, these interfaces are currently more challenging to build than traditional user interfaces. Following are some of the conceptual, methodological, and technical challenges that TUI developers face:

Interlinked virtual and physical worlds While conventional interfaces rely on virtual objects only, TUIs use both virtual and physical objects, which coexist and exchange information with each other. An important role of the TUI developer is to determine which information is best represented digitally and which is best represented physically [2].

Multiple behaviors In graphical user interfaces (GUIs), each widget encapsulates its behavior. In a TUI, the behavior of a physical object is not determined by the nature of the physical object alone, but also by that object's interactions with other physical and virtual artifacts. Furthermore, the object's behavior may change when a new physical object is added to the TUI. Therefore, when specifying the behavior of a certain physical object, the designer is required to take into consideration the mutual impact of physical objects.

Multiple actions Foley et al. [3], in a widely accepted paper, suggested that, in an interactive graphical system, there are six fundamental interaction tasks: select, position, orient, path, quantify, and text. However, in a three-dimensional, physical world, there are numerous activities that can be performed with, or upon, any physical object (e.g., squeeze, stroke, toss, push, tap, pat, etc.). Hence, the designer is charged with selecting and defining which are the meaningful actions.

No standard input/output devices In TUIs, there are currently no standard input or output devices for accomplishing a given task. For example, measuring the movement of an object may be implemented using magnet sensation, RFID, or computer vision. Though identical in purpose, each technology currently requires a different set of physical devices, instructions, and code.

O. Shaer (✉) · N. Leland · R. J. K. Jacob
Department of Computer Science, Tufts University,
Medford, MA 02155, USA
E-mail: oshaer@cs.tufts.edu

E. H. Calvillo-Gamez
Universidad Politécnica de San Luis Potosí,
San Luis Potosí, Mexico

Thus, the integration of novel technologies into an application is difficult and costly [4].

Continuous interaction Tangible user interfaces support a combination of discrete and continuous interaction. When users continuously interact with physical objects, they perceive that their motions are directly mapped to changes in the digital information. However, existing event-based models for designing interactive systems currently fail to capture continuous interaction explicitly [5]. Thus, TUI software developers are often required to deal with continuous interaction in considerably ad-hoc, low-level programming approaches.

Distributed interaction Dourish [6] noted that, in a TUI, there is no single point of interaction, as multiple users can simultaneously interact with multiple physical objects. In addition, the same action in a given interaction may be distributed across multiple physical objects. Existing models for designing interactive systems usually handle multiple input devices by serializing all input into one common stream [5]. However, in TUIs, this method is less appropriate since the input is logically parallel and the users' perception is that two or more dialogs are taking place simultaneously. It is important to note that we are referring here to the importance of parallel design at the conceptual and software model level, and not at the microprocessor level (which may deal with such an interface in a parallel or single channel way).

To address these challenges, a software toolkit for specifying, simulating, and building TUIs is needed. However, before such a toolkit can be built, it is first necessary to identify the set of constructs required to describe the structure and functionality of a large subset of TUIs. Our proposed token and constraints (TAC) paradigm provides a set of core constructs, which are, for a wide range of TUIs, what widgets and events are to GUIs.

The paper is organized as follows. We first discuss related work, which lays the foundation for the TAC paradigm. We then introduce the TAC paradigm and explain how it should be applied in specifying TUIs. Next, we evaluate the TAC paradigm by testing its ability to describe a wide variety of existing TUIs. Following the TAC paradigm evaluation, we discuss its contribution to TUI developers. Finally, we discuss our conclusions and plans for developing a new toolkit based on our approach.

1.1 Related work

Fitzmaurice et al. [7] laid the foundation for a new framework with their discussion of the graspable user interface. In 1997, Ishii and Ullmer [1] suggested the term "tangible user interface" as referring to systems that augment the real world by coupling digital information to tangible objects. Holmquist et al. [8] introduced taxonomy of physical objects that can be linked to digital information, suggesting three categories of objects: containers, tokens, and tools. By their description, containers refer to generic objects used to move information between platforms, tokens refer to physical objects used to access stored information and tools are used to manipulate information. Koleva et al. [9] suggested a framework for the classification of TUIs based on the degree of coherence between physical and digital objects.

In his dissertation, Ullmer [2] introduced the concept of token + constraint systems, which considers tokens as physical objects representing digital information or operations, and constraints confining regions in which tokens are placed and manipulated. He suggested that the design space of TUIs includes three high-level categories: interactive surfaces, which are systems where the user manipulates physical objects upon a planar surface; constructive assemblies, which refers to systems inspired by Lego, in which users interconnect modular elements; and tokens + constraint. This classification of TUIs provides a basis for considering TUIs as related elements of a larger design space rather than isolated systems; however, it does not cover the entire TUI design space. The TAC paradigm uses Ullmer's [2] token + constraint approach as its basis. It then extends the concept of constraint, stressing that a TUI may be described as a set of relationships between a token, a set of constraints, and a variable. The main principles of the TAC paradigm were introduced in [10]. A new, extensively revised version of the TAC paradigm is discussed in detail in the following section.

Several human–computer interaction (HCI) models are relevant to the modeling of tangible interaction and the structure of TUIs. The MVC model [11] highlights the separation of a GUI into a *view*, provided by the graphical display, *control*, provided by the mouse and keyboard, and *model*. Taking MVC as their basis, Ullmer and Ishii [2, 12] presented an interaction model for TUIs, the MCRit, that highlights the integration of representation and control in TUIs. PAC [13] is an implementation model that recursively structures an interactive application in terms of presentation, abstraction, and control. Myers [14] suggested a model for handling input that encapsulates interactive behaviors into a few interactor object types. Application programmers can then create instances of these interactor objects.

Finally, a few models and toolkits have appeared in related research areas that link the physical and digital worlds. Fishkin et al. [15] proposed a paradigm and design framework for embodied user interfaces. Jacob et al. [5] presented a software model and specification language for non-WIMP user interfaces. The term "non-WIMP user interface" refers to a set of emerging computer environments, such as virtual environments, eye-movement-based user interfaces, physical and ubiquitous computing. Their approach was based on the view that the essence of non-WIMP dialog is a set of continuous relationships, most of them temporary. VRID [16] is a design methodology for developing VR interfaces. iStuff [17], Phidgets [18], and Papier-Mâché [4] are all toolkits providing high-level APIs for different sensing mechanisms. iStuff is designed specifically for the ubiquitous computing environment and supports wireless devices, Phidgets encapsulates communication with USB-attached physical devices, and Papier-Mâché intends to support computer vision, RFID, and Barcode. While all of these facilitate the integration of physical objects and sensing mechanisms into TUIs, they do not provide support for the association of high-level semantics to physical objects. Building upon this foundation, we have extended these ideas and we propose a paradigm for specifying TUIs.

1.2 The Marble Answering Machine

One of the earliest illustrations of interlinking the physical and digital worlds is provided in the design of the Marble answering machine. It was designed and prototyped by Durrell Bishop, while a student at the Royal College of Art, as a way to explore ways in which computing can be taken off the desk and integrated into every day objects [19].

We have selected the Marble Answering Machine [19] as a leading example throughout this paper because it clearly demonstrates the concept of accessing digital information by manipulating physical objects. In the Marble Answering Machine, incoming voice messages are attached to marbles. To play a message, the user grabs a message (marble) and places it in an indentation on the machine [19]. To return a call, the user places a marble in an indentation in an augmented telephone. Though additional functionality is available, this describes the functionality necessary to understand our examples. Figs. 4, 5, 6, 7, and 8 illustrate the Marble answering machine.

2 The TAC paradigm

The TAC paradigm we propose identifies the common components and properties sufficient for specifying the structure and functionality of a wide range of TUIs.

Our approach is based on the notion that a TUI may be described as a set of relationships between physical objects and digital information. These relationships are defined by the TUI developer and may be instantiated by the user. After a relationship has been instantiated, a user may manipulate physical objects in order to access or manipulate digital information.

As is common in evolving research areas, the terminology used to discuss TUIs has not yet reached widespread consensus. Therefore, we would like to begin by defining the following terms: *pyfo*, *token*, *constraint*, *variable*, and *TAC*. After defining these terms, we will use them to describe TUIs. The Marble Answering Machine [19] will be used as an example throughout both the introduction of the TAC paradigm terminology and the ensuing discussion of the TAC paradigm properties.

2.1 TAC terminology

A *pyfo* is a physical object that takes part in a TUI. A pyfo may be comprised of a number of other pyfos (e.g., a box may be made up of six pyfo ‘‘sides’’). We chose the term ‘‘pyfo’’, which has a Spanish influence, in order to avoid the use of the term ‘‘Object’’, which has multiple meanings in the field of computer science and HCI. We also wanted to avoid the term ‘‘physical object’’ because of its common use in reference to elements of the physical world, which have no connection to TUIs. The term ‘‘pyfo’’ has the advantage of being brief while still bearing resemblance to ‘‘physical object’’, and maintaining its specificity to the world of TUIs.

Pyfos may enhance their physical properties with digital properties such as graphics and sound. In the Marble answering machine [19], both the marbles and the answering machine itself are considered pyfos. There are two types of pyfos: tokens and constraints. Each pyfo can be a token, a constraint, or both.

A *token* is a graspable pyfo that represents digital information or a computational function in an application. The user interacts with the token in order to access or manipulate the digital information.

The physical properties of a token may reflect the nature of either the information or the function it represents. Also, the token’s physical properties may afford how it is to be manipulated. For example, we consider marbles in the Marble answering machine [19] tokens. The user interacts with a marble in order to access the message it represents. The physical properties of the marble suggest that the user can grab the marble and pick it up. WebStickers is another example of a system where users take advantage of an object’s physical properties, and use these properties as cognitive cues for finding Web sites [20]. In WebStickers, the user couples a certain Web site to a physical object, and then interacts with the object to access the Web site. We consider the physical objects as tokens.

A *constraint* is a pyfo that limits the behavior of the token with which it is associated. The physical properties of the constraint guide the user in understanding how to manipulate the token and how to interpret the compositions of token and constraints. The constraint limits the token’s behavior in the following three ways:

1. The physical properties of the *constraint*, such as orientation, material, textures, etc., suggest to the user how to manipulate (and how not to manipulate) the associated *token*. For example, in the Marble Answering Machine [19], the size and shape of the ‘‘play message’’ indentation affords placement of a marble in the indentation. The size of the indentation

also suggests that only one marble at a time may be placed in the indentation (see Fig. 1).

2. The *constraint* limits the physical interaction space of the *token*. When *tokens* are manipulated within the confines of a *constraint*, their interaction space is limited to the space provided by the *constraint*. For example, in the Senseboard system [21], pucks are manipulated within the confines of a grid. Alternatively, additional *pyfos* placed within the confines of the same *constraint* serve to further limit the *token's* interaction space. In Senseboard for example, existing pucks on the Senseboard grid prevent users from placing a new puck in the exact same location as an existing puck (see Fig. 2).
3. The *constraint* serves as a reference frame for the interpretation of *token* and *constraint* compositions. Compositions of *token* and *constraints* may be interpreted either in spatial terms, such as coordinates or numerical values, or in relative terms, such as the prepositions: *first*, *left*, *of*, *beside*, etc. [2]. In both cases, the compositions are interpreted with respect to a reference frame.

A *reference frame* is defined in physics as a way of assigning coordinates to a given space or as a way of describing positions in space [22]. In the TUI context, spatial and relative relationships between a *token* and a *constraint* are expressed with respect to the *constraint*. Therefore, the *constraint* provides the reference frame for the spatial or relative interpretation. A *constraint* also provides the reference frame for interpreting compositions of a *token* and some other *pyfo* associated with the same *constraint*, thus, sharing the same reference frame (see Fig. 3). We have already discussed why we consider this "other *pyfo*" as a *constraint*. Hence, a single *constraint* provides the reference frame for interpreting the relationship between a *token* and all of its *constraints*.

An example of *token* and *constraint* compositions that are relatively interpreted can be found in the next section under the discussion of Tangible Query Interfaces [23]. Figures 1, 2, and 3 illustrate the three ways *constraints* can limit the behavior of a *token*.

A *variable* is digital information, or a computational function, in an application. Some variables are coupled to *tokens*, while others are semantic *variables* in the application.

A *TAC* is the relationship between a *token*, its *variable*, and one or more *constraints*. Often, this relationship is temporary. The relationship is defined by the designer, and is instantiated by either the designer or the user. The physical manipulation of a TAC is the manipulation of a *token* with respect to its *constraints*, and it has computational implications.

For example, the composition of a marble located in the incoming message queue is considered as a TAC. The TAC consists of the following: the marble is a *token*, the indentation (representing the incoming message queue) and the other marbles in the queue are the *constraints*, and the message coupled to the marble is the *variable*. Having defined a terminology, we now use this terminology to describe TUIs.

Fig. 1 The physical properties of the *constraint* affords placement of the *token* in the indentation

2.2 Properties of the TAC paradigm

The TAC paradigm contains five key properties; *couple*, *relative definition*, *association*, *computational interpretation*, and *manipulation*. Following the definition of these properties, we evaluate their robustness by demonstrating their application on a wide variety of TUIs. We use the Marble Answering Machine [19] as a leading example throughout the definitions, describing how the system would be designed using the TAC paradigm.

Couple: a pyfo must be coupled with a variable in order to be considered a token When a *pyfo* is coupled with a

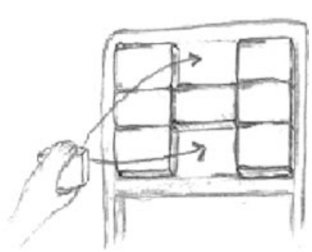

Fig. 2 The board and other pucks physically limit the interaction space of the new puck

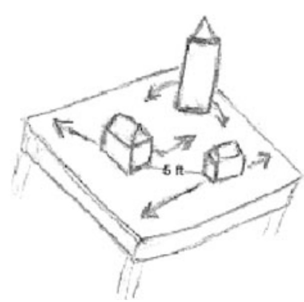

Fig. 3 The table serves as a reference frame for the user's interaction with the buildings

variable, it becomes a *token*. The designer defines what type of variable may be associated with a certain *pyfo*. The actual coupling of the *variable* to the *token* is then executed either by the designer at design time or by the user at run time. The former is referred to as static coupling, the latter as dynamic binding.

For example, in the Marble Answering Machine [19], a marble coupled to a message is considered as a *token*. When designing the system, the designer would determine that a "message" *variable* should be associated with a marble. Then, at run time, the machine couples incoming messages to specific marbles.

Relative definition: each pyfo may be defined as a token, a constraint, or both. The marbles in the Marble Answering Machine [19] are coupled to incoming messages and, therefore, can be considered as tokens. However, consider marble A; it is constrained by the indentation of the incoming message queue of the machine, which physically channels the marble. It is, however, also constrained by the other marbles in the message queue, which limit the amount of available space in the queue. In this case, a marble may be defined as either a *token* or a *constraint*, depending on the marble being discussed.

The first and second properties are illustrated in Figs. 4 and 5.

Association A new TAC is created when a token is physically associated with a constraint. New *constraints* may be added to an existing TAC. When a *token* is associated with a *constraint*, a new TAC relationship is created. In the example of the Marble answering machine [19], when marble A is associated with the incoming message queue, a new TAC is created. The new TAC consists of a *token*—marble A, and a *constraint* list that includes the queue itself and the other marbles in the queue. Later, when a new marble is added to the queue, the new marble is added as a *constraint* to the TAC containing marble A as a *token*. When the user removes marble A from the queue, this TAC is destroyed.

Token and *constraints* have a recursive structure in that a given TAC can serve as a *token* or a *constraint* for

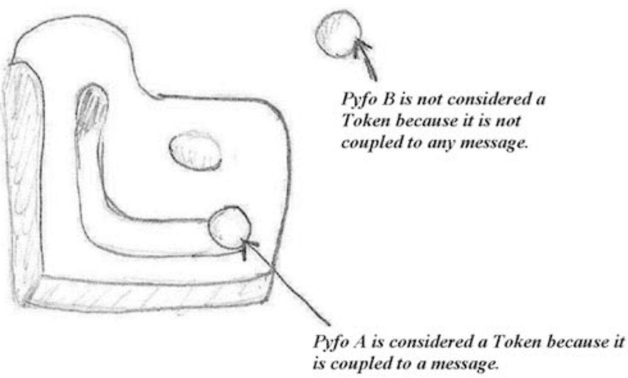

Fig. 4 Couple (redrawn based on Durrell Bishop's illustration from [19])

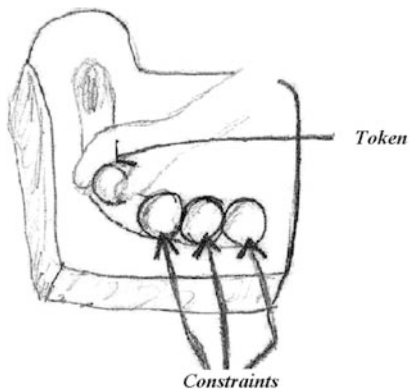

Fig. 5 Relative definition (redrawn based on Durrell Bishop's illustration from [19])

other TACs; that is, larger TAC structures subsume smaller TAC instances. An example of a recursive TAC is discussed in the next section under the specification of Tangible Query Interfaces [23].

Figure 6 illustrates the third property.

Computational interpretation: the physical manipulation of a TAC has computational interpretation. The manipulation of a *token* with respect to its *constraints* has computational interpretation and, therefore, changes the state of the application. Manipulation of a *token* outside its *constraints* has no computational interpretation. Only when a *token* is associated with *constraints* does its manipulation have computational interpretation.

For example, in the case of the Marble Answering Machine [19], when the user adds a marble to the replay indentation, the machine plays the message; when the marble is removed from the indentation, the message stops playing and the message status is changed from new to old. The user observes the change in message

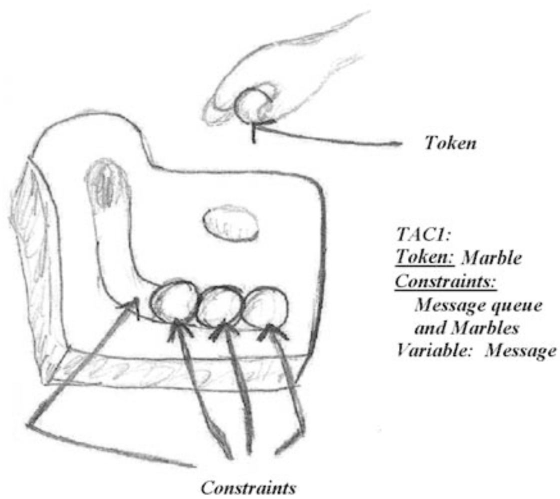

Fig. 6 Association (redrawn based on Durrell Bishop's illustration from [19])

state in the form of feedback from the application. The designer determines the nature of this feedback.

On the other hand, the manipulation of a marble located outside the defined regions of the machine (its constraint) has no computational interpretation and, therefore, has no effect on the application's state.

Manipulation: each TAC can be manipulated discretely, continuously, or in both ways. The physical manipulation of a token is afforded by the physical properties of its constraints. Consider the example of the Marble Answering Machine [19]. The TAC, comprised of the marble *token* and the replay indentation *constraint*, can be manipulated by adding or removing the marble to/from the indentation. This is a discrete manipulation, which can be derived from the physical properties of the indentation, in that the size and the location of the replay indentation suggest to the user that the TAC may be manipulated in these ways. An example of continuous behavior can be found later in this paper under the Urp [24] specification.

Figures 7 and 8 illustrate the fourth and fifth properties.

2.3 Specifying a TUI using the TAC paradigm

The TAC paradigm describes a TUI as a set of TAC relationships. Specifying a TUI using the TAC paradigm consists of defining the possible TAC relationships within a TUI. For each TAC, the developer defines its *token* and *constraints*. He then describes the behavior of a TAC by specifying the actions that may be performed on its *token*, together with their responses. The TAC relationships are defined by the developer, but may be instantiated by either the user or the developer. Typically, instantiation of a TAC is initiated in response to a discrete event. For each TAC that may be instantiated or destroyed at run time, the TUI developer must define the discrete actions *add* and *remove*, which instantiate or destroy the TAC. These actions may also have additional computational effects on the TUI beyond simply instantiating and destroying TACs. An example of this is found in the Urp [24] specification.

In this section, we have presented a conceptual framework for TUIs. We have introduced a terminology and key properties, which together provide TUI developers with a common vocabulary and conceptual tools for specifying the functionality and structure of TUIs. We identified *pyfos*, *tokens*, and *constraints* as the core components of TUIs, and suggested that TAC objects are the conceptual building blocks of TUIs. Similar to widgets in a GUI, TAC objects encapsulate the set of meaningful actions that can be performed upon a physical object in a TUI. In the next section, we evaluate the TAC paradigm's ability to describe a broad range of interfaces by specifying a variety of TUIs using the proposed paradigm.

3 Evaluating the TAC paradigm

We have proposed a new paradigm. In order to evaluate its ability to specify a large subset of TUIs, we have specified a wide range of existing TUIs using the constructs provided by the paradigm. Each of the TUI specifications we have chosen to include in this paper serves as a representative for an existing class of TUIs. Together, they cover an important and large subset of the TUI design space. In our selection of TUI classes, we utilized Ullmer's [2] division of the TUI design space into three high-level classifications and selected representatives from each classification: interactive surfaces, constructive assemblies, and token + constraint. We also selected TUIs that fall outside these classifications.

We specified each TUI by listing its TAC relationships in terms of *representation* and *behavior*. *Representation* refers to the physical association of a *token* to its *constraints*. *Behavior* refers to the variable coupled to the token and to the actions a user may perform on a TAC. For each TAC that can be instantiated and destroyed at run time, we specified the discrete actions *add* (add *token* to *constraint*) and *remove* (remove *token* from *con-*

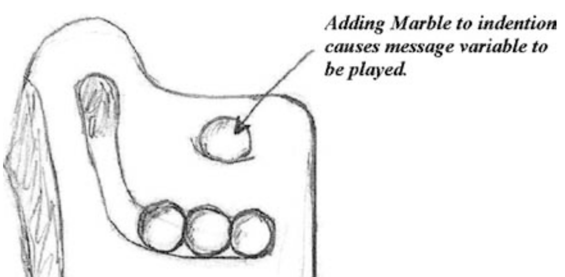

Fig. 7 Computational interpretation (redrawn based on Durrell Bishop's illustration from [19])

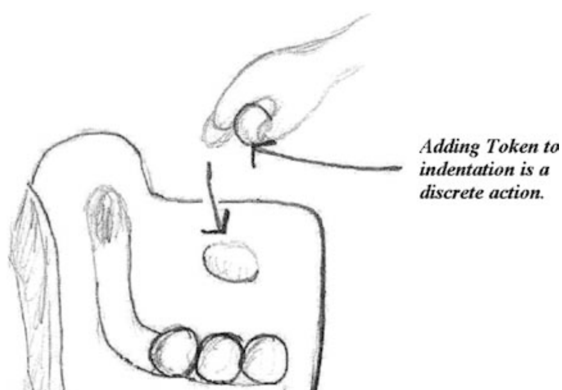

Fig. 8 Manipulation (redrawn based on Durrell Bishop's illustration from [19])

straint), which, correspondingly, activate or deactivate a TAC.

3.1 The Designers' Outpost

The Designers' Outpost [25] is representative of the interactive surfaces category. The system allows users to design the information architecture of websites. Working collaboratively on a whiteboard where regular Post-It notes represent Web pages, users are able to structure the information by moving notes around the board and then link the information and annotate it using electronic pens. Table 1 summarizes the specification of the Designers' Outpost system.

An interesting aspect of the Designers' Outpost specification is that the scope of the system extends beyond TUIs. Once the physical note becomes an electronic note, it is no longer a physical object, and therefore, the system moves away from a state that is purely tangible. Instead, the system operates more like a GUI where the user directly manipulates an electronic note. However, as seen below, the TAC paradigm is still able to describe the system and is therefore, applicable for systems that combine tangible and graphical interaction.

3.2 Computational building blocks

Computational building blocks [26] is representative of the constructive assemblies category. It allows users to construct a structure using Lego-type building blocks, which is later displayed graphically on a computer screen. Each block is encoded with information about its shape, color, and texture. Also, each block is aware of those blocks that it considers as neighbors. A neighbor is any block physically connected on the top or bottom of a given block.

An interesting aspect of this system specification is that any given block may be considered as either a *token* or a *constraint*, depending on the TAC relationship being discussed. A block is considered a *token* when the block is statically coupled to a digital block variable and physically constrained by its neighbors (the blocks to which it is connected). This same block is also considered a *constraint* for its neighbors. Table 2 summarizes the specification of the Computational Building Blocks system.

3.3 Tangible Query Interfaces

Tangible Query Interfaces [23] is representative of the token + constraint category. It uses physically constrained tokens to express, manipulate, and visualize parameterized database queries. We would like to highlight two interesting aspects which arise from the specification of the tangible query interfaces system. First, the system illustrates the recursive structure of a TAC. TAC 3 in Table 3 is comprised of a token, a parameter bar with upper and lower sliders, constrained by a query rack. A closer look at the token in TAC 3, reveals that the *token* itself is comprised of two TACs; TAC 1 and TAC 2 in Table 3. Since a TAC can be comprised of other TACs, we defined the structure of TACs as recursive. Second, in this system, we see an example of relative interpretation of *token* and *constraint* compositions. The proximity of the parameter bars

Table 1 The Designers' Outpost [25] specification using the TAC paradigm

TAC	Representation		Behavior		
	Token	Constraints	Variable	Action	Observed feedback
1	Paper note	BoardNotes	Paper note	Add to board	Adds paper note to board. Add a Web page to a Web page list
				Remove from board	Removes paper note from board and removes any links to it
				Move	Moves the physical location of paper note maintaining any links
				Tap	Activates emenu
2	Eraser	BoardPaper notesLinks	Links	Add to board	None
				Remove from board	None
				Erase	Removes links
3	Eraser	BoardPaper notesDrawings	Drawings	Add to board	None
				Remove from board	None
				Erase	Remove drawings
4	Move tool	BoardNotesEnote	Enote	Add to board	None
				Remove from board	None
				Move	Moves the physical location of enote maintaining any links
5	Pen	BoardNotes	Link	Add to board	None
				Remove from board	None
				Draw	Adds links
6	Pen	Board	Drawing	Add to board	None
				Remove from board	None
				Draw	Adds drawings

Table 2 Computational Building Blocks [26] specification using the TAC paradigm

TAC	Representation		Behavior		
	Token	Constraints	Variable	Action	Observed feedback
1	Block	Neighbor blocks	Block	Add to neighbor blocks Remove from neighbor blocks	Adds block to digital structure Removes block from digital structure

located on the query rack impacts the system's interpretation of the query parameters. That is, if one unit is adjacent to, or "next to" another, the operator AND is applied to the two adjacent parameters. When units stand separate in the query rack (and there is more than one parameter), the OR operator is applied. The query rack serves as a reference frame for both the user's interpretation and the actual computational interpretation of the parameter bars and query rack composition. Table 3 summarizes the specification of the Tangible Query Interfaces System.

3.4 ComTouch

ComTouch [27] is a TUI providing the user with tangible feedback initiated by a remote user. The system is designed like a normal cell phone, but it augments verbal communication with haptic feedback, allowing the user to express a non-verbal message, such as emotion or conversional cue. When user A is talking to user B, he may squeeze the cell phone to augment his verbal message. User B then feels the vibration of his device. The intensity of the vibration can be interpreted as the intensity of message. It is easy to see that ComTouch does not fall under any of Ullmer's classifications. Table 4 presents the ComTouch specification.

The specification of this system consists of two TAC relationships. In both relationships, the hand is considered as a physical object, and serves as either a *token* or a *constraint*. The first TAC considers the cell phone as a *token* linked to the *variable*, *non-verbal message*. The vibration of the cell phone allows a user to receive this message and to interpret it. The vibration is constrained by the hand, which provides the reference frame for its interpretation. In the second TAC, the hand is now considered as a *token* linked to the *non-verbal message*. The user presses the hand on the cell phone to express the message. The user's ability to press is physically constrained by the cell phone.

Table 3 Tangible query interfaces [23] specification using the TAC paradigm

TAC	Representation		Behavior		
	Token	Constraints	Variable	Action	Observed feedback
1	Upper slider	Parameter bars Lower slider	Upper bound variable value in query	Slide (vertically)	Display is updated to reflect new upper bound
2	Lower slider	Parameter bars Upper slider	Lower bound variable value in query	Slide (vertically)	Display is updated to reflect new lower bound
3	Parameter bar and sliders (TAC 1 and TAC 2)	Query rack Other parameters	Query	Add to query rack	Adds a new parameter to the query. Display is updated accordingly
				Remove from query rack	Removes a parameter from the query. Display is updated accordingly
				Slide (horizontally)	Display is updated according to the applied logical operator, AND or OR

Table 4 ComTouch [27] specification using the TAC Paradigm

TAC	Representation		Behavior		
	Token	Constraints	Variable	Action	Observed feedback
1	Cell phone	Hand	Non-verbal message	Add to hand Remove from hand Vibrate	None None None
2	Hand	Cell phone	Non-verbal message	Add to cell phone Remove from cell phone Press	None None Remote phone vibrates

3.5 Urp

The Urp [24] system is also an example of an interactive surfaces interface. It uses physical models of buildings manipulated on a table to help urban planners perform analysis of shadows, proximities, reflections, wind, and visual space. Table 5 summarizes the specification of the Urp system. The specification of this system highlights two interesting features of TUIs; continuous interaction and temporary relationships between *token* and *constraints*.

The manipulation of buildings upon the table surface illustrates continuous interaction. Using the example of a building's shadow, when the user slides a building from point A to point B, the system does not wait until the building gets to point B to display the changes in the shadow. Instead, the shadow cast by the building is continuously updated to reflect each position of the building between point A and B.

The temporary nature of the TAC relationship is demonstrated by TAC 6, the TAC comprised of the material tool as a *token* and a building as a *constraint*. The instantiation and manipulation of this TAC are one and the same. The moment the user touches the building with the material tool, the TAC is activated and the material of the building is changed. The relationship only lasts for a moment. Once activated, the TAC may be immediately terminated, since there is no further need to hold the material tool against the building.

4 Discussion

The TAC paradigm was intended to provide a simple and elegant set of constructs sufficient to describe the functionality and structure of a broad range of TUIs. These constructs in turn will serve as the basis for a high-level description language and a software toolkit for TUIs.

To evaluate the TAC paradigm's ability to describe a broad range of TUIs, we analyzed a wide variety of interfaces and have shown that the set of constructs it

Table 5 Urp [24] specification using the TAC paradigm

TAC	Representation		Behavior		
	Token	Constraints	Variable	Action	Observed feedback
1	Building	TableOther buildings Roads	Building	Add to table	Displays shadows cast by the building, according to time of day
				Remove from table	Removes physical building from display; removes any display information related to the building
				Move	Moves the physical location of the building, updating the display accordingly
2	Road tool	Table Buildings	Road	Add to table	Adds a road to the display with simulated traffic on it
				Remove from table	Removes road and any associated traffic lights from display
				Slide	Moves the physical location of the road, adjusting traffic and traffic lights accordingly
3	Distance measuring tool	TableBuildings Roads	Distance function	Add to a building or a road	Distance tool has two ends. When distance end is added to one constraint, a drag line appears; if added to two constraints, a line connecting the two constraints appears showing the distance between the two. Otherwise, if erase end is used, the line disappears from display
				Remove from a building or a road	None
4	Wind tool	Table	Wind	Add to table	Airflow simulator activated
				Remove from table	Airflow simulator deactivated
				Orienting	Display reflects changes in wind direction
5	Anemometer tool	Table	Wind	Add to table	Display indicates wind flow magnitude at the arrow point
				Remove from table	Display of wind flow magnitude removed
6	Material Transforming tool	BuildingsTable	Building	Add to a building	Changes the building's material. Display changes accordingly
				Remove from a building	None
7	Clock dial	Clock board	Sun	Set time	Display changes to reflect new time of day
8	Clock (TAC 7)	Table	Sun	Add to table	None (enables time setting)
				Remove from table	None (disables time setting)

provides is sufficient for specifying TUIs classified as interactive surfaces, constructive assemblies, and token + constraints systems [2], as well as additional interfaces we studied outside these classifications, such as ComTouch [27]. We believe that the TAC paradigm may be applicable to specify an even broader range of interfaces, and, to test this assumption, we intend to use the TAC paradigm in the development of new TUIs.

The TAC paradigm addresses the conceptual challenges discussed in the introduction. The notion of a TAC allows designers to encapsulate the token's behavior at the TAC level rather than at the pyfo level. Therefore, the TUI developer can specify the set of actions that are meaningful when executed with respect to a certain set of constraints. For example, in the Tangible Query Interfaces system [23], sliding a parameter bar is an action that only has meaning when manipulated with respect to the query rack. By allowing the encapsulation of actions in the TAC, our paradigm mediates the challenges of *multiple actions* and *multiple behaviors*.

The simplicity of specifying a TUI using the TAC paradigm may encourage TUI developers to better address the challenges of *interlinked virtual and physical worlds* by experimenting with representing *tokens* in different forms, either digital or physical. The TAC paradigm implicitly supports distributed interaction, as it is simply a declarative specification for a set of TAC relationships maintained in parallel.

To address technical challenges such as *lack of standard input/output devices* and *continuous interaction*, a toolkit providing developers with a set of practical tools is needed. Such a toolkit is discussed under future work.

The TAC paradigm currently does not provide a mechanism or a language for describing and/or analyzing issues such as form and the affordance of different materials, colors, or shapes. Currently, it only addresses the function and the structure of TUIs. However, it is expected that a toolkit based on the TAC paradigm would provide designers with the mechanism to experiment with *pyfos* in different forms.

As TUIs evolve, the importance of discussing and analytically analyzing and comparing alternative designs for TUIs increases. The TAC paradigm itself is not meant to be an analytical tool for analyzing or comparing design hypotheses; rather, it is concerned with identifying the constructs necessary for a TUI toolkit. However, it may serve as a basis for the development of an analytical tool aimed at assisting designers in gaining new insights in the TUI design process.

5 Future work

Having established a conceptual framework for specifying TUIs, we are currently developing a high-level description language and software toolkit to bridge the gap between the conceptual foundations of TUIs and the practical complexities of building these systems.

Our toolkit will allow designers to specify a TUI using a high-level description language based on the TAC paradigm. This specification would then be translated into a simulation program or a program controlling a set of physical interaction objects. With our toolkit, the TUI implementation will consist of two parts: a lexical handler handling the communication of the user with a set of physical objects, and the application logic. A lexical handler is provided for each implementation mechanism supported by the toolkit so the same application logic may be prototyped using different implementation mechanisms. A control component is responsible for the communication between the lexical handler and the application logic, thus, providing desirable technological independence.

We have built a prototype toolkit providing designers with a 3D graphical modeling tool and form-based tools to specify TUIs. The system translates the TUI description into a high-level description language and simulates tangible interaction in a Java3D-based virtual reality environment. We intend to develop a full toolkit for specifying, simulating, and programming TUIs. We are developing an automatic generator of interactive C code from a high-level specification which supports TUI prototyping using a Handyboard microcontroller. We are also developing a lexical handler that supports prototyping interfaces using RFID tags. We are interested in cooperating with existing physical computing toolkits to extend the technologies supported by our system.

6 Conclusions

We have presented the TAC paradigm, a conceptual framework for TUIs. Our paradigm is based on the notion that a TUI consists of a set of TAC relationships, some of which are recursive and/or temporary. We evaluated the proposed paradigm by applying its key properties to a wide variety of TUIs, and showed that the set of constructs it provides are indeed sufficient for describing a wide variety of TUIs.

Many concerns have been left for future consideration. Matters such as interoperability of TUIs, mass production of TUIs, security and privacy in tangible interactions are all potential research directions. We look forward to collaborating with others to explore the exciting space of TUIs.

Acknowledgements We would like to thank Mark D. Gross, Ellen Yi-Luen Do, Ken Camarata, and Brygg Ullmer as well as the reviewers and editors for their valuable feedback on this paper. We would also like to thank the students of the HCI research group at Tufts University and the students of the Design Machine Group at the University of Washington for discussing these ideas. We want to extend thanks to Tufts students Shan Lu and Cris Barbero for working with us, and to Juana Maria Hernandez from the Universidad Politecnica de San Luis Potosi. Finally, we thank Hiroshi Ishii and the Tangible Media Group at MIT. This work was partially supported by a grant from the National Science Foundation Partnerships for Innovation program.

References

1. Ishii H, Ullmer B (1997) Tangible bits: towards seamless interfaces between people, bits and atoms. In: Proceedings of the ACM conference on human factors in computing systems (CHI 1997), Atlanta, Georgia, March 1997
2. Ullmer B (2002) Tangible interfaces for manipulating aggregates of digital information. PhD thesis, Massachusetts Institute of Technology
3. Foley JD, Wallace VW, Chan P (1984) The human factors of computer graphics interaction techniques. IEEE Comput Graph Appl 4(11):13–48
4. Klemmer SR, Li J, Lin J, Landay JA (2004) Papier-Mâché: toolkit support for tangible interaction. In: Proceedings of the ACM conference on human factors in computing systems (CHI 2004), Vienna, Austria, April 2004
5. Jacob RJK, Deligiannidis L, Morrison S (1999) A software model and specification language for non-WIMP user interface. ACM Trans Comput Hum Interact 6(1):1–46
6. Dourish P (2001) Where the action is: the foundations of embodied interaction. MIT Press, Cambridge
7. Fitzmaurice GW, Ishii H, Buxton W (1995) Bricks: laying the foundations for graspable user interfaces. In: Proceedings of the ACM conference on human factors in computing systems (CHI'95), Denver, Colorado, May 1995
8. Holmquist LE, Redström J, Ljungstrand P (1999) Token-based access to digital information. In: Proceedings of the first international symposium on handheld and ubiquitous computing (HUC'99), Karlsruhe, Germany, September 1999
9. Koleva B, Benford S, Hui Ng K, Rodden T (2003) A framework for tangible user interfaces. In: Proceedings of the real world user interfaces workshop at the 5th international symposium on human–computer interaction with mobile devices and services (MobileHCI 2003), Udine, Italy, September 2003
10. Calvillo-Gámez EH, Leland N, Shaer O, Jacob RJK (2003) The TAC paradigm: a unified conceptual framework to represent tangible user interfaces. In: Proceedings of the 1st Latin American conference on human computer interaction (CLIHC 2003), Rio de Janeiro, Brazil, August 2003
11. Burbeck S (1987) Applications programming in Smalltalk-80: how to use model-view-controller. Available at http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html
12. Ullmer B, Ishii H (2000) Emerging frameworks for tangible user interfaces. IBM Syst 39(3–4):915–931
13. Coutaz J (1987) PAC: an implementation model for dialog design. In: Proceedings of the 2nd IFIP international conference on human–computer interaction (INTERACT'87), Stuttgart, Germany, September 1987
14. Myers BA (1990) A new model for handling input. ACM Trans Inf Syst 8(3):289–320
15. Fishkin K, Moran T, Harrison B (1998) Embodied user interfaces: towards invisible user interfaces. In: Proceedings of the 7th international conference on engineering for human–computer interaction (EHCI'98), Heraklion, Crete, Greece, September 1998
16. Tanriverdi V, Jacob RJK (2001) VRID: a design model and methodology for developing virtual reality interfaces. In: Proceedings of the ACM symposium on virtual reality software and technology (VRST 2001), Banff, Canada, November 2001
17. Ballagas R, Ringel M, Stone M, Borchers J (2003) iStuff: a physical user interface toolkit for ubiquitous computing environments. In: Proceedings of the ACM conference on human factors in computing systems (CHI 2003), Fort Lauderdale, Florida, April 2003
18. Greenberg S, Fitchett C (2001) Phidgets: easy development of physical interfaces through physical widgets. In: Proceedings of ACM symposium on user interface software and technology (UIST 2001), Orlando, Florida, November 2001
19. Crampton Smith G (1995) The hand that rocks the cradle. In: ID magazine May/June 1995, pp 60–65
20. Ljungstrand P, Redström J, Holmquist LE (2000) WebStickers: using physical tokens to access, manage and share bookmarks to the web. In: Proceedings of the conference on designing augmented reality environments (DARE 2000), Elsinore, Denmark, April 2000
21. Jacob RJK, Ishii H, Pangaro G, Patten J (2002) A tangible interface for organizing information using a grid. In: Proceedings of the ACM conference on human factors in computing systems (CHI 2002), Minneapolis, Minnesota, April 2002
22. Merriam-Webster online dictionary (2003) Available at http://www.m-w.com
23. Ullmer B, Ishii H, Jacob R.J.K (2003) Tangible query interfaces: physically constrained tokens for manipulating database queries. In: Proceedings of the 9th IFIP international conference on human–computer interaction (INTERACT 2003), Zurich, Switzerland, September 2003
24. Underkoffler J, Ishii H (1999) Urp: a luminous-tangible workbench for urban planning and design. In: Proceedings of the ACM conference on human factors in computing systems (CHI'99), Pittsburgh, Pennsylvania, May 1999
25. Klemmer SR, Newman MW, Farrell R, Bilezikjian M, Landay JA (2001) The Designers' Outpost: a tangible interface for collaborative web site design. In: Proceedings of the international symposium on user interface software and technology (UIST 2001), Orlando, Florida, November 2001
26. Anderson D, Frankel JL, Marks J, Agarwala A, Beardsley P, Hodgins J, Leigh D, Ryall K, Sullivan E, Yedidia J (2000) Tangible interaction + graphical interpretation: a new approach to 3D modeling. In: Proceedings of the international conference on computer graphics and interactive techniques (SIGGRAPH 2000), New Orleans, Louisiana, July 2000
27. Chang A, O'Modhrain S, Jacob RJK, Gunther E, Ishii H (2002) ComTouch: design of a vibrotactile communication device. In: Proceedings of the conference on designing interactive systems (DIS 2002), London, June 2002