# TUIMS: Laying the Foundations for a Tangible User Interface Management System

**Nancy Leland**
Department of Computer Science
Tufts University
nleland@cs.tufts.edu

**Orit Shaer**
Department of Computer Science
Tufts University
oshaer@cs.tufts.edu

**Robert J.K. Jacob**
Department of Computer Science
Tufts University
jacob@cs.tufts.edu

## ABSTRACT

This paper lays the foundations for the development of a Tangible User Interface Management System (TUIMS). It presents a paradigm for representing TUIs, the TAC paradigm, which identifies the core components of TUIs. Building upon the TAC paradigm, it introduces TUIML, a high-level description language for TUIs. The concept of TUIMS is proposed and a built proof of concept prototype is discussed.

## KEYWORDS

Tangible User Interface, Token and Constraints (TAC), User Interface Management System (UIMS), User Interface Description Language(UIDL).

## INTRODUCTION

In the last decade we have 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 referred to as Tangible User Interfaces [6] (TUIs).

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.

TUI designers face unique conceptual, methodological and technical challenges. Some of these challenges are: mapping digital information to physical objects, specifying the relationships between physical objects, describing the tangible interaction dialogue and dealing with the lack of standard technologies for implementing TUIs.

To address these challenges we propose the concept of a Tangible User Interface Management System (TUIMS) draws from earlier work on UIMS[4]. TUIMS allows designers to specify tangible interaction in a high level description language (TUIDL). This specification would then be either automatically or semi-automatically translated into a graphical simulator or a program controlling a set of physical interaction objects.

## THE TAC PARADIGM

The Token and Constraints (TAC) [5] paradigm is a unified conceptual framework for 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 designer and may be instantiated by the user. After a relationship has been instantiated, a user may manipulate the physical objects in order to access or manipulate digital information.

As is common in evolving research areas, the terminology used to discuss tangible user interfaces has not yet reached widespread consensus. Therefore we would like to begin by defining the following terms:

**A Pyfo** is a physical object that takes part in a TUI. There are two types of Pyfos: Tokens and Constraints.

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

**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 manipulating the associated token and interpreting the compositions of a token and a set of constraints.

**A Variable** is a data object, or a computational function in an application.

**A TAC (Token And Constraints)** is a relationship between a token, a variable and a set of constraints. The physical manipulation of a TAC is the manipulation of a token in respect to a set of constraints, and it has computational implications.

The TAC Paradigm contains five key properties:

**Couple,** A Pyfo must be coupled with a variable in order to be considered a token.

**Relative Definition,** Each Pyfo may be defined as a token, a constraint or both.

**Association,** A new TAC is created when a token is physically associated with a constraint. New constraints may be added to an existing TAC.

**Computational Interpretation,** The physical manipulation of a TAC has computational interpretation.

**Manipulation,** Each TAC can be manipulated discretely, continuously or in both ways.

In order to evaluate the ability of the TAC Paradigm to describe a full range of TUIs we used the TAC paradigm to

specify existing TUIs. We selected a variety of TUIs that covers the familiar TUI design space and showed that the TAC Paradigm may be easily used to specify these interfaces. A description of the selected TUIs and their specifications may be found in [5].

## HIGH LEVEL DESCRIPTION LANGUAGE FOR TUIS

TUIML (Tangible User Interface Markup Language) is a high level description language for TUIs. It draws from two main foundations: the TAC Paradigm [5] and model based user interfaces [9]. In order to provide an effective TUIDL, the TUIML design satisfies the following requirements: a comprehensive design process support, a repository of design data, representation of both abstract and concrete aspects of TUIs, the use of XML as an underlying technology.

TUIML predefines five basic components: **Task, Domain, Representation, TAC** and **Control.** The task and domain components describe the semantics of the TUI. The representation and TAC components describe the syntax of the TUI system. The representation component defines a set of logical physical objects, the TAC component defines the context for interaction actions performed upon these logical physical objects and determines which semantic functions are invoked as a result of an interaction action. These components do not specify the TUI implementation mechanism. The Control component keeps track of the TUI system state during run time and maps lexical level (implementation mechanism) events to the syntactic level interaction actions thus provides desirable technology independence. Following is a portion of the TAC model describes the Marble Answering Machine[3].

```
<TAC_MODEL id='tacm1'>
  <TAC_ELEMENT ID='tac1' NAME='Marble in replay indentation' >
        <RELATION_STATEMENT DEF='is_token' REF='r1'/>
        <RELATION_STATEMENT DEF='is constraint' REF='r2'/>
        <MANIPULATION_ELEMENT ID='Construct'>
          <RELATION_STATEMENT DEF='invokes' REF='t1'/>
        </MANIPULATION_ELEMENT >
  </TAC_ELEMENT>
</TAC_MODEL>
```

In order to validate the expressiveness and usefulness of TUIML, we undertook a number of validation activities include: Hand coded representation of new and existing TUIs.

## TUI MANAGEMENT SYSTEM

A TUIMS allows designers to specify TUIs using TUIDL. This specification would then either be automatically or semi-automatically translated into a graphical simulator or a program controlling a set of physical interaction objects. With a TUIMS an interactive application consists of two parts: a lexical handler handling the communication with the user and an application component containing the application logic. The TUIMS Dialogue Manager component is responsible for the communication between these two components.

We built a prototype TUIMS which provide designers a 3d graphical modeling tool and form based tools to specify TUIs (see figure 1). The system translates the TUI description into a TUIML representation and simulates the tangible interaction in a Java3D based VR environment. We use this prototype in the development of a new TUI.

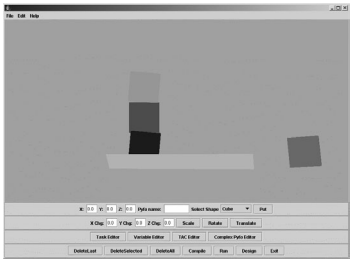

Figure 1, A TUIMS prototype is used to graphically simulate a TUI

## FUTURE WORK

We intend to continue developing our prototype TUIMS into of a full TUIMS for specifying, programming and testing TUIs . We are currently developing an automatic generator of interactive C code from TUIML specification which supports TUI prototyping using a Handyboard microcontroller. We are also looking forward to cooperate with existing physical toolkits such as iStuff[1] and Papier-Mâché [8] to extend the technologies supported by the TUIMS.

## CONCLUSION

In this paper we have presented the concept of a TUIMS and laid the foundation for its development. We presented the TAC paradigm which identifies the core components of TUIs. Building upon the TAC paradigm, we presented TUIML, a high-level description language for TUIs. Finally, we discussed the TUIMS concept which was used in building a prototype TUIMS.

## REFERENCES


1. Anderson Ballagas, R., Ringel, M, Stone, M., Borchers, J, "iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments", CHI 2003
2. B.Ullmer, H. Ishii, and R.J.K. Jacob, "Tangible Query Interfaces: Physically Constrained Tokens for Manipulating Database Queries," *Proc. INTERACT 2003 Conference*, 2003.
3. Crampton Smith, G. "The Hand That Rocks the Cradle." I.D., May/June 1995, pp. 60-65.
4. D. Olsen. "User Interface Management Systems: Models and Algorithms." Morgan Kaufmann, San Mateo, CA, 1992.
5. E.H. Calvillo Gamez, N. Leland, O. Shaer, and R.J.K. Jacob, "The TAC Paradigm: Unified Conceptual

Framework to Represent Tangible User Interfaces," *CLIHC 2003 Latin American Conference on Human-Computer Interaction*, 2003.

6. 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.

7. Klemmer S.R, "Papier-Mâché: Toolkit support for tangible interaction." in *UIST 2003* Doctoral Consortium.

8. P. Szekely. “Retrospective and challenges for model-based interface development.” In F. Bodart and J. Vanderdonckt, editors, *Computer Aided Design of User Interfaces (CADUI’96)*, pages 1–27, Wien, 1996. Springer-Verlag.