Tangible Agile Mapping: Ad-hoc Tangible User Interaction Definition

James A. Walsh, Stewart von Itzstein and Bruce H. Thomas
School of Computer and Information Science
University of South Australia
Mawson Lakes Boulevard, Mawson Lakes, South Australia, 5095
james.walsh@setoreaustralia.com, stewart.vonitzstein@unisa.edu.au, bruce.thomas@unisa.edu.au

Abstract

People naturally externalize mental systems through physical objects to leverage their spatial intelligence. The advent of tangible user interfaces has allowed human computer interaction to utilize these skills. However, current systems must be written from scratch and designed for a specific purpose, thus meaning end users cannot extend or repurpose the system. This paper presents Tangible Agile Mapping, our architecture to address this problem by allowing tangible systems to be defined ad-hoc. Our architecture addresses the tangible ad-hoc definition of objects, properties and rules to support tangible interactions. This paper also describes Spatial Augmented Reality TAM as an implementation of this architecture that utilizes a projector-camera setup combined with gesture-based navigation to allow users to create tangible systems from scratch. Results of a user study show that the architecture and our implementation are effective in allowing users to develop tangible systems, even for users with little computing or tangible experience.

Keywords: Tangible user interfaces, programming by demonstration, organic users interfaces, proxemic interactions, authoring by interaction.

1 Introduction

All of us utilize the physical affordances of everyday objects to convey additional information about some mental 'cognitive system', as a means of reducing errors compared to when we simulate the system mentally (Myers, 1992). If a teacher were explaining the reactions between different chemicals, they would pick up different objects to represent elements, moving them closer together to indicate a reaction, changing the chemicals' state. Despite the simple rules for these interactions, complex configurations can easily be created. Mentally tracking these roles and states, however, introduces a cognitive overhead for both primary and collaborative users. Tangible User Interfaces (TUIs) help this problem by designing a system that utilizes the affordances of physical objects. However, despite research into TUIs existing for some years, the adoption of TUIs is only recently being seen in the consumer market. For example, the Sifteos commercial product followed on from the Siftables research project (Merrill et al., 2007).

Rapid reconfiguration of workspaces is required in many tasks, for example to facilitate different users' work flows and task switching (Fitzmaurice, 1996). The ability for a user to fully customize their system is difficult to achieve. By allowing the system to be at least partially (re)defined during use, the system can compensate for user diversity. The user already has the knowledge regarding how they want to interact with the system and what they want it to do. However, despite research into TUIs and customization, no systems currently exist that support the tangible ad-hoc definition of objects and functionality.

Currently, designing and using tangible and augmented systems involves three main steps; *calibration* of the system, *authoring* the content (both models and logic), and *interacting* with the system. This process does not support a natural workflow, where objects, roles and functionality need to rapidly change in an ad-hoc nature. *This paper presents our investigations into the merging of the authoring and interacting stages to create a form of Authoring By Interaction (ABI), where system content and functionality can be defined through normal methods of system interaction.*

The architecture described in this paper, called Tangible Agile Mapping (TAM), works towards ABI, enabling previously unknown objects to be introduced into the system during the interaction phase. Through these interactions, users can introduce new objects and define their properties and functionality. This allows the authoring of new virtual content and systems, using only normal interactions with the system – lowering the threshold for developing TUIs. This allows novice users to develop TUI applications in the same vain GUI toolkits enabled the development of desktop user interfaces (UIs) by a wider audience. TAM also allows for tangible systems created by users to be saved for future reuse and distribution. This research is driven by the questions:

- *How can the system enable users to easily develop TUIs when the system has no context of what the user is trying achieve?*

- *How can the system support definition of these interfaces inside the environment when there are no existing UI* *components outside the tangible realm (i.e. no mouse, keyboard or IDE)?*

Whilst there exists systems that investigate interactions with ad-hoc objects, the separating factor and key contribution of this work is its focus as a primary means of enabling ad-hoc UI and functionality definition in a tangible manner. The implementation of the architecture as an example system supports the design of a tangible chemistry set and basic tangible war game, amongst others. To the authors' knowledge, this generalized, program-as-you-go approach to TUIs offers a new application, outside previous works' focus on application specific development.

This paper makes the following contributions:

- An architecture to support the conceptual model of tangible ABI systems based on previous literature and pilot study.
- A functioning system demonstrating this architecture, incorporating an encapsulating UI.
- The evaluation of the functioning system through the results of its implementation and a user study.

As the focus of this work is on the ad-hoc development of systems in a purely tangible manner, this work does not purport to contribute or focus on the visual tracking, gesture interaction, or programming by demonstration fields. As such, this work makes the assumptions that such a system has six degree-of-free (6DOF) tracking and object mesh generation capabilities, ideally enabling the formation of organic user interfaces (OUI) (Holman and Vertegaal, 2008). Recent advances illustrate that these are not unreasonable assumptions (Izadi et al., 2011).

The remainder of the paper is structured as follows: related work is discussed, identifying a number of challenges. A pilot study conducted to evaluate how users would interact with such a system is described, which precedes a description of an architecture to support ad-hoc tangible interaction. The implementation of this architecture is explored in an example system and evaluated through a user study with regards to the success and user experience. We then conclude with future work and final thoughts.

2 Related Work

Our research follows previous work in HCI, more specifically TUIs and OUIs, as well as sharing similarities with programming by demonstration/example systems.

TUIs afford physical objects, spaces and surfaces as a coupled interface to digital information (Ishii and Ullmer, 1997). The Bricks system (Fitzmaurice et al., 1995) explored graspable UIs, as a predecessor to TUIs, enabling manipulation of digital elements through 6DOF tracked 'bricks', exploring the advantages of spatial-based interaction. The affordances of graspable UIs provided a number of advantages in bimanual and parallel interaction, utilization of spatial reasoning, externalization of interfaces for 'direct' interactions, and support for collaborative workspaces (Fitzmaurice et al., 1995).

The MetaDESK (Ullmer and Ishii, 1997) explored rendering maps based on the location of miniature building surrogates, with appropriate warping of the map to ensure correct map alignment. Alongside this work, the authors suggested a number of conceptual equivalencies between traditional GUIs and TUIs.

URP (Underkoffler and Ishii, 1999) and other similar projects explored tangible spatial and environmental configuration and interaction. The use of physical props enabled the user to control system variables through tangible interaction. URP allowed users to quickly experiment with different configurations, by experimenting with different spatial layouts of structures. These systems supported bi-directional communication regarding system configuration, allowing the user to receive feedback regarding their interactions.

Tangible Tiles (Waldner et al., 2006) allowed interaction with projected elements using gestures ('scooping' them up and 'sliding' them off). Of note was that users could create copies of the digital content. Rekimoto and Saitoh (1999) explored UI inheritance as not being intuitive for novice users, leading to the question of whether a TUI should utilize shallow or deep copies when managing virtual properties. Travers' (1994) direct manipulation system, supported both shallow and deep object copies, noting that whilst inheritance can have a high pedagogical value, it can cause issues for users who have no concept of inheritance.

Ullmer (2002) explored the GUI model-view-controller (MVC) equivalency in TUIs, identifying three distinguishing categories of TUIs; interactive surfaces, constructive assemblies, and tokens+constraint (TAC). The TAC category allowed constraints to be imposed on the TUI based on the physical constraints of the object, utilizing their natural affordances for logical constraint.

Holman and Vertegaal (2008) introduced OUIs as non-planar displays that are the primary means of both output and input, allowing them to 'become' the data they are displaying. This follows closely with the real world with little distinction between input and output, with perhaps the closest equivalent being cause and effect (Sharlin et al., 2004).

Papier-Mâché (Klemmer et al., 2004) explored the abstraction of sensor fusion to provide generic inputs to the system, allowing programs to be developed without managing low-level input. In a similar line, Kjeldsen et al. (2003) abstracted vision based inputs. Applications requiring inputs would ask the middleware for a specific input (such as a button), which is dynamically generated and mapped by the system based on the available inputs. More recently, the Proximity Toolkit (Marquardt et al., 2011) abstracted multi-device hardware to provide a set of program events for proxemic interactions.

Both VoodooIO (Villar et al., 2006) and Phidgets (Greenberg and Fitchett, 2001) explored reconfigurable physical toolkits, supporting rapid development via plug-and-play hardware. Similarly, the iRos/iStuff (Borchers et al., 2002) system provided a patch-panel framework for functionality. Despite offering reconfiguration, the systems only looked at mapping controls.

Bill Buxton coined the term Programming By Example (PBE) as systems that require the user to specify every system state (Myers, 1986), allowing the user to work through a specific example of the problem. Halbert (1984) characterized them as "do what I did", whereas inferential Programming By Demonstration (PBD) (Dey et al., 2004) systems are "do what I mean". However, inferential systems create procedures that are both complex and unstructured (Myers, 1986). Myers (1986) noted that PBD/PBE systems must provide support (even implicitly) for conditional and iterative operations. Whilst you can only demonstrate one branch at a time, it was noted that demonstrational interfaces would be appropriate in scenarios where users possess high level domain knowledge that could be represented using low level commands repeatedly or in an interface with limited options that the user wants to customize. Following the impact of GUI toolkits, visual programming systems allowed non-programmers to create moderately complex programs with minimal knowledge (Halbert, 1984).

Hacker (1994) explored the psychology of tangible problem solving and task completion through goal attainment using action regulation theory. Following this, since pragmatic interactions can reveal new information (Kirsh and Maglio, 1994), system interactions should enable trial-and-error with a low cost of speculative exploration (Sharlin et al., 2004).

As highlighted, this work builds on the concepts present in a number of different fields. TAM explores the application of PBE to TUIs, building on preceding work in HCI and GUI design, abstraction and interaction. Despite work on abstracting interactions, developing the interactions and content is still isolated from the use of the system. Through TAM, a number of these fields are brought together in the hope of enabling ABI.

3 Derived Challenges

Despite the previous research on tangible user interfaces, digital augmentation as well as PBD/PBE systems, a number of problems still exist for TUI developers and users alike, creating significant scope for further research. This creates a number of derived challenges:

1. Tangible systems must be *designed specifically for their application*. There is no generic architecture for developing tangible systems available for developers, which in turn makes few systems available to users.
2. *Augmented tangible systems involve a number of sub-components*: high level object tracking and spatial relationships, utilizing 3D models of the objects for augmentation and the logic for managing the actual interactions. These all must be either developed from scratch or heavily re-worked to support ad-hoc functionality (the Proximity Toolkit did however start to explore proxemic abstraction).
3. Most importantly, *tangible systems are not accessible to end users* and cannot be customized beyond their original purpose despite a clear benefit.

4 Exploratory Interview

Following early discussions regarding the development of a system to support ABI, exploratory interviews were conducted with six participants to gain a better understanding of how users think about interacting with tangible systems, as well as how they would envision extending them. A tangible version of the Fox, Duck and Grain game was used to explore how users, in an ideal scenario, would interact and communicate such a game to the system for tangible replication. The game involves two banks of a river, a boat and a fox, duck and a bag of grain. Users must get all the items to the other river bank, without leaving the fox alone with the duck or the duck alone with the grain. The boat can only carry one object at a time. This game was chosen as it involves a number of key concepts:

- Defining object roles;
- Defining common groups/types;
- Defining physical and quantitative restraints;
- Defining virtual entities/regions (for use as the river banks);
- Defining interactions both between individual objects and groups of objects, as well as interacting with virtual objects (regions).

In an ideal world, the user could convey such a system to the computer as if it were another person. However, the user will always have to restate their problem definition in a structure and language that the computer understands. This creates two problems, the first is having the user reformulate their problem to match the system's structure, with the second being the transfer of this knowledge into the system, creating two points of failure where the user is managing the problem in two different cognitive structures. To explore this, the questions discussed in the study were:

1. *How would users ideally like to communicate instructions for a tangible system to the computer?* This involved participants physically and vocally describing the system, step-by-step;
2. *What kind of interactions do users expect the system to be able to support?* This involved having participants actually 'playing-out' this game in a tangible sense;
3. *How should the user communicate with the system to perform instructions for 'learning' versus 'playing'?* This involved having the participants explain how they would like to communicate a change in task focus.

The interview was conducted with six people (two female, four male), two of which had a background in computer science.

Participants separated interactions into two groups; good (legal) and bad (illegal) moves. Participants would program an interaction, e.g. leaving the fox and duck alone on the riverbank, and identify that as being a 'bad' interaction, asking the system to highlight the objects in red to convey the error state. However, when programming a 'good' interaction, they wanted a different operation, even though the only difference is that a 'good' operation highlights in green. They did not intuitively abstract this operation to *'here is the interaction', 'here is the output'*. One non-technical participant did note that they wanted to generalize an interaction to support the substitution of objects.

Whilst programming interactions into the system, most users created regions for 'accepted' and 'unaccepted' interactions. To program rules into the system, users would move objects next to one another in the appropriate regions. The remainder wanted the system to project an 'accepted/unaccepted' or 'yes/no' menu next to the objects.

For identifying objects, users preferred text names/abbreviations and/or colours. However, the use of colour was then overloaded by the fact that participants suggested its use to define groups/types of objects, which could be defined based on proximity, drawing a line around them or holding all the objects in their hands.

For feedback about incorrect moves, all but one participant wanted feedback to be local to the offending interaction. The other wanted the whole system to provide feedback, referring to "tilting a pinball machine". A different participant (one without a computer science background) wanted a log or system tray so that they could keep track of their interaction history to see "what worked and what didn't".

Physical constraints for an object's location were defined by pointing or drawing a line, then moving the object along that path. Quantitative constraints (i.e. only one object allowed on the boat) were defined by performing the interaction, then writing the legal number of objects near the boat.

Most participants chose the use of a virtual button to switch tasks to program rules or start playing the game, with the remainder wanting to use a thumbs-up gesture.

The final component of the interview involved asking the participants if their expectations of such a system would be addressed based on our initial thoughts regarding such a system. This involved verbally describing the system whilst using props and drawing aids. All participants agreed that our proposed task-flow supported their internal processes regarding how they would conduct it.

5 Tangible Agile Mapping

The Tangible Agile Mapping (TAM) system directly addresses the derived challenges and incorporates feedback from the interviews. One of the primary contributions of this paper stems from the theoretical architecture to enable ad-hoc object definition and interaction. The following sections describe the architecture, as well as the implementation of that architecture and the application to manage it.

5.1 Architecture

To enable a flexible, ad-hoc environment, a certain level of complexity is required within the architecture to enable adaptation. Any TUI system that wants to enable ad-hoc functionality will need to support the following functions at a high level:

- *Definition of core objects*, which could be either physical or virtual.
- *Define types/groups* of objects that enable substitution.
- *Properties* that can be used to describe those objects.
- *Support associations* between those properties (including many-to-many associations).
- *Define rules* for those objects which in-turn can make any number of changes to objects.
- Support *sequential interactions.*

It is important to realize that this complexity is hidden from the user, as they are working at a higher level. Using these functions, there are four different scenarios that can occur:

	Isolated Updates	**Common Updates**
Isolated interactions	Scenario 1 *e.g. "two different interactions affecting two different objects"*	Scenario 2 *e.g. "two different interactions affecting the same objects"*
Overlapping interactions	Scenario 3 *e.g. "two different interactions involving some of the same objects, but affecting two different objects"*	Scenario 4 e.g. "*two different interactions involving some of the same objects and affecting the same objects"*

Table 1: Interaction scenarios

Despite Scenario 1 being the primary method of interaction, the system needs to support all four scenarios.

To address these requirements, our architecture consists of six classes (Figure 1) to define the model component of Ullmer's TUI equivalent to the MVC. Our implementation, to be described later, follows this architecture, as we believe the features described in this architecture are core to any tangible ad-hoc system. The remainder of this section describes our architectural support for the defined high-level functions.

5.1.1 Definition of Core Objects

All objects are defined using InteractionObjects (InObjs), which are further described by Properties. These InObjs trigger *Actions* (rules) as part of an *Interaction*, which performs a copying from/to Properties as defined by a *PropertyMap.* We also require a core application to update the system.

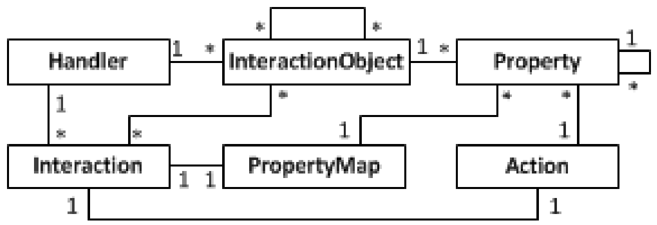

Figure 1: Relationships between the core components of the architecture

We use an inheritance structure for representing objects and properties. To interact with both physical objects and digital systems, we use a core InObj class. The class serves as the base that all objects (physical or virtual) inherit from. This class provides basic capabilities such as system names, IDs, tracking information, references to properties as well as providing virtual functions for drawing and managing the object's properties.

5.1.2 Define types/groups

Each InObj also contains two sets of references to InObjs, one for listing groups to which this InObj is a *member*, the other, to define other InObjs which are a *group member* of this object. As such, InObjs can be grouped in a hierarchical manner. A group object on its own is a single, non-physical InObj with child members; however each of those child InObj members may have their own children, creating a tree. This means any interaction occurring with an InObj can easily be substituted by any other InObj (including those acting as groups), allowing any of those objects to act as a substitute. This is crucial when defining generalized interactions for substitution. For example, when defining interactions for a game's playing pieces, you do not want to specify the same rules for all playing pieces. You just create a group (i.e. a new InObj) containing all playing pieces, and create rules based on that group. The use of groups enables the definition of types for both physical and virtual objects.

5.1.3 Properties

To describe objects, we can apply any number of Properties. All properties extend from a base Property, which (like InObj) constitutes a collection of sub-properties in addition to possibly holding its own data. For example, a location Property may constitute Position and Orientation Properties. This allows Properties to be created in a hierarchical nature. To capture an object's current configuration, Property needs to support the creation of deep-copies for later retrieval. This is used to capture the before state of an object when performing 'toggled' interactions. To enable deep-copies, all properties must provide a method for returning a new Property of the same type.

Defining Rules and Associations

To enable interactivity, TAM manages interactions through a set of objects for each interaction. Interaction tracks the Action (rule set) that triggers the interaction; based on a set of InObjs (stored in Interaction) that can trigger it, as well as what properties need to be updated to what value. Interactions essentially contain a set of rules (defined by an Action), what objects can break them (stored in Interaction) and the changes that occur (stored in a PropertyMap, discussed later). The Action class takes a set of InObjs from Interaction and evaluates if the Action has been triggered and what objects triggered it, and is overridden by each specific Action (rule) supported by the system. This is important as when members of a group trigger an action, we need to know which one(s) were involved (i.e. which playing piece out of all playing pieces actually triggered it). This function is over-ridden by action-specific handlers (e.g. proximity, orientation, etc.). Multi-Action interactions (e.g. based on location and orientation) are managed as two separate Interactions (for location and orientation) which serve as prerequisites to a single Interaction that performs the actual property updates.

To handle updating Properties, Interaction stores a PropertyMap. PropertyMap is responsible for defining a set of 'From' and 'To' Properties, defining where object values need to be retrieved from and which objects to copy to. To support complex mappings, PropertyMap also contains a *MappingAction* to define how to handle non-1:1 mappings. PropertyMap also has Push/PopHistory functions, which capture and store a deep copy of all Properties involved for later retrieval (as is needed for toggled interactions).

5.1.4 Sequential Interactions

To support updates to common properties, i.e. scenarios 2 and 4 in Table 1, where the rules dictate that the same property must have two different values, Interaction contains a set of references to other Interactions that must occur as pre-requisites (either simultaneously or at any time previously). As a result, we dictate the precedence for the order of execution of interactions, as well as enabling staged interactions, i.e. A must occur before B.

We believe this architecture is sufficiently flexible to support a wide range of complex interactions, beyond simple TUIs. For example, whilst not explored in this paper, the architecture does support functionality such as the user editing core system components and functions as part of their interactions, as discussed in the next section.

5.2 Implementation

The implementation of the architecture was tailored towards its use in Spatial Augmented Reality (SAR), and was aptly titled SAR-TAM. The implementation extended InObjs into *PhysicalInteractionObject* (PhInObj) and *VirtualInteractionObject* (ViInObj). PhInObj offers attributes for the object's location and bounding box functionality (as handled internally by the system). The application of ViInObjs allows the system to interact with existing virtual systems and remains as future work – however we envisage little-to-no modification will be needed.

One of the core features of TAM is the ability to assign any existing property to any other existing property (e.g. you can easily assign an object's colour based on its position). As such, the implementation only has single 'core' property (i.e. a property that actually stores data), *IntProperty*, which stores a single integer value. All other properties such as Colour, Position, Orientation, Outline, etc. all consist of a generic property that has IntProperties as children. Sub-Properties may override a *Draw()* method that actually 'applies' the property to the system. For example, in this implementation ColourProperty has four IntProperty children (for the RGBA colour channels). Colour's Draw() function calls OpenGL's glColor() function, passing the values of the four IntProperty children. The hierarchical nature means any property can encapsulate any other property, e.g. the Location property just contains Position and Orientation properties, which are themselves collections of IntProperties.

In our implementation, PropertyMap contains multiple *PropertyMapping* objects, each containing the individual one-to-one mappings for Properties as a result of an interaction. By default when assigning a property, the system attempts to do a linear assignment of properties to copy from/to, i.e. Colour 1's Red will be copied to Colour 2'd Red, etc (based on their order in the set). In the case of mismatches (i.e. mapping three elements to a single element), the PropertyMapping being applied has an attribute to handle such situations, e.g. use the max/min or averaged values. Values can also be assigned as relative to the current value or absolute, as well we being capped at either/or a max/min value or operate in a modulus fashion – this allows for interactions over a variably defined range, beyond simple boolean interactions. Whilst this many-to-one functionality exists in a working manner architecturally, modifying this attribute has not been explored in the TAM UI and remains as future work.

Action provides two evaluation functions, *EvaluatingIncludingGroups* and *EvaluateExcludingGroups*, used to evaluate either with or without objects' children/group members. Currently, only one type of action is supported in TAM, ProximityAction. Action support will grow in the future to include orientation and virtual actions (to support input from existing digital systems) as well as temporal based interactions.

Whilst not explored in this paper, SAR-TAM does support the possibility of the user editing core system functions as part of their interactions. One example could be the use of physical objects (e.g. blocks) to create custom keyboards. Using blocks, the user could define an interaction where upon placing a block on a keyboard key, the key becomes 'virtually glued' to the block, allowing the position of the system-managed, virtual entity to be modified at run-time using a custom interaction defined by the user. The user could also create interactions to change the letters on the keys, creating custom keyboards and layouts for other languages. One suggested use by someone trialling the system was to create a tangible video editor so that film editing can leverage the user's spatial skills.

6 System Overview

SAR-TAM uses two Microsoft Kinects and an OptiTrack 6DOF system (which enables object tracking using retroreflective markers and infrared light). One Kinect is downward facing and is used for object detection and enabling touch interactions. The other Kinect faces the user for skeleton tracking to enable gesture input. The OptiTrack is used to track objects once they have been registered with the system, as the Kinect cannot reliably track objects between frames whilst be handled.

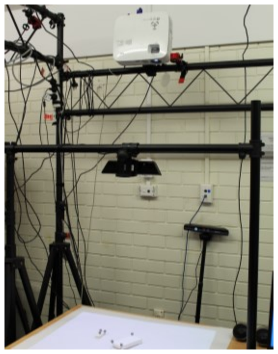

Figure 2: The SAR-TAM tabletop with projector, tabletop and pose Kinects and OptiTrack system (red cameras)

The system runs at 20fps due to delays in updating both Kinects and detecting the user's skeleton contour. SAR-TAM utilises a state machine, with poses (static gestures) the primary method of navigation, allowing users to start with a 'blank slate' with no system-specific input tools.

All poses inherit from a base pose, which returns a boolean based on a skeleton provided by the OpenNI framework. Users must hold a posture for at least 500ms to help prevent false positives. Feedback for pose detection is provided using a graphical component on the table showing the user's contour with a skeleton overlay. Skeletal bones with less than 70% accuracy are rendered in red. Upon performing a pose, the user's contour colour is changed to white, instead of the previously randomly assigned pastel colour. A description of the matched pose is displayed underneath.

6.1 Using the System

To use the system, users interact solely through visual and audio cues. The projected UI is designed to be minimal (Figure 3a) to allow users to develop completely custom, immersive systems. The current state (either 'Interacting', 'Introducing Object', 'Defining Group' or 'Defining Interaction') is displayed on the top of the display area as well as a brief set of instructions for the current step. Upon changing states, instructions are updated on the display and read aloud using a text-to-speech engine.

To introduce objects, users place an object on the surface and perform a 'one arm open' gesture, as if to say "here is a new object" (Figure 3a). The system then prompts users for an object name, and highlights the outline of the object detected each frame with the Kinect (Figure 3b). Users enter a name using a simplified projected keyboard displayed at a fixed location in front of the user.

Upon pressing 'Confirm' (replacing the 'Enter' button) users then select a default colour for the object from a 'linear' style colour chart (Figure 3c). The object is augmented using the object's contour and projecting the colour. Given contours update each frame, they are subject to jitter.

When the desired colour is selected, users place their forearms vertically parallel as if they were about to take a photo to 'capture' the current configuration. This is known as the *Confirm* pose. Any objects that have been defined are now shown in their selected colour, with the name projected along side. New objects can be introduced at any stage.

Once an object is formally introduced, the system starts to match objects between the Kinect and OptiTrack system each frame. Each PIO is attached to a single OptiTrack marker. Every frame, TAM locates the markers and matches the contours detected by the Kinect, allowing objects to be tracked frame-by-frame. Currently, this is based on a simple proximity test.

Once users have defined at least two objects, they can either create groups/sets of objects (to enable object substitution using groups in Interactions) or define an interaction rule set. To define a group, users extend both arms out towards the table as if to gesture, "here are a group of objects". The system then projects a virtual keyboard, and asks users to point to objects in the group and enter a name for the group. Tracked objects are now only identified by their projected name until users 'point' at them by placing their finger within 5 cm of the object, at which point the object is highlighted using its default colour. Users then enter a group name and press 'Confirm'. Creation of the group is confirmed by voice prompt.

To create a set of rules for an interaction, users place their forearms at 90° (Figure 4a). The system then prompts users to point to objects involved in the interaction and perform the Confirm pose. Should any of those objects be members of a group, the system will prompt users to resolve this ambiguity by displaying a menu next to each object with group memberships. The menu shows the object's name as the default option, with the object's groups' names as options (Figure 4c). Users are prompted to select which group the object can be substituted by and perform the Confirm pose. The system then prompts users to perform the interaction (at the moment this is limited to arranging objects based on proximity relative to each other) and then perform the Confirm pose. The system plays the sound of a SLR camera taking a photo to provide feedback that the arrangement was 'captured'. The system then prompts users to highlight which objects change during that interaction and presents the colour chart, allowing selected objects to have their colour changed. Once this is done, users perform the Confirm pose and the system goes back to the normal state, allowing users to trigger interactions or continue defining new objects/groups/rules.

7 Evaluation

Regardless of the level of flexibility offered by an architecture, there will always remain a level of adaptation imposed on the user due to individual differences; however we seek to minimize this. As such, for a user to be able to use TAM, there are two things that must occur for the user to fully externalize their internal thoughts into a tangible system:

1. Users must adapt their view/architecture of system to match that supported by the adaptive system.
2. Users must translate that knowledge into the system.

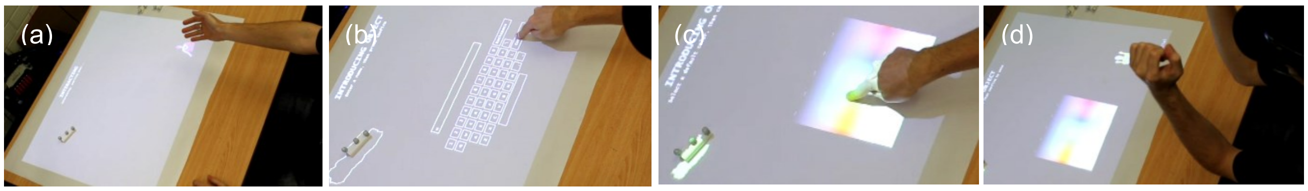

Figure 3: Introducing a new object: (a) Performing the introduction pose, (b) entering a name, (c) selecting a default colour and (d) confirming the selection.

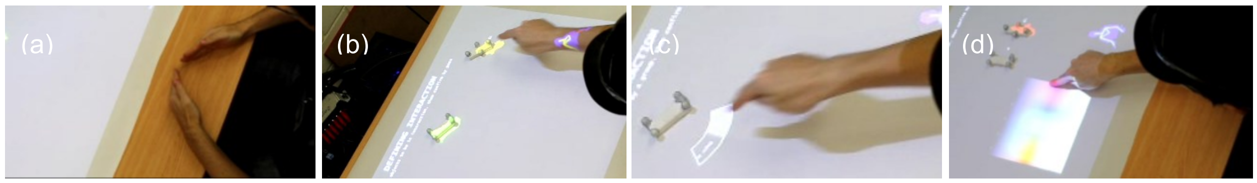

Figure 4: Creating rules for an interaction: (a) Performing the new interaction pose, (b) selecting objects involved, (c) resolving group selection for substitution and (d) performing the changes as a result of the interaction

As a result, our evaluation was designed to evaluate;

- How easily can users grasp the concepts involved in TAM to convert a scenario into the required structure?
- How easily can users then communicate that structure to the system?

This was evaluated by means of an exploratory user study. We employ an experimental design similar to Scott et al. (2005) to gain an understanding of how users engage with and understand the system. The study consisted of the participants first being seated at a desk, watching a video that explained the system and demonstrated a single, group-based interaction between three objects to simulate the reaction between hydrogen and chlorine to form hydrogen-chloride (two hydrogen atoms were defined, and a group created so either hydrogen could trigger the reaction). To ensure equal training, all participants could only watch the video once, but could pause and ask questions at any time.

After the video, participants were asked to create three scenarios, each scenario extending the last, to demonstrate a minimal game. The tasks consisted of:

1. A country that was being invaded by an attacking army. Upon the attacking army reaching the country, they would change its national colour to red.
2. Same as Task 1 except with a defending army. Upon both the defending and attacking armies reaching the country, the defending army defeats the attacking and changes the national colour to green in mourning of those lost.
3. Same as Task 2, but with two defending and two attacking armies. No matter which combination reach the country (as long as there is at least one attacking and one defending), the attacking army is defeated and the colour changed to green in remembrance of those lost.

The country and armies were represented by numerous large, white pencil erasers. Upon the participant moving the objects beyond the participant-defined limit, the system would trigger the associated change in colour. This allowed the user to physical move the different armies and see the resulting state of the country object.

Participants were provided with a reference sheet showing each pose and its corresponding function in the system. This was provided as we were not interested in evaluating the particular poses, rather the functionality that they linked to. Participants were read aloud from a copy of each task, and then given a printed copy. All participants were video recorded. At the conclusion, participants filled out a questionnaire focusing on the system's intuitiveness and ease of use. The questionnaire was a mix of visual analogue scales and qualitative questions. The questions focussed on how easy, intuitive and appropriate they found each sub-section of the system to function (introducing objects, defining groups, etc.). They were also asked about the level of guidance provided by the system and how the system functioned versus how they would explain the problem to another person. The study concluded with an informal discussion.

7.1 Results

The user study conducted consisted of 21 participants (2 female, 19 male, mean age of 24), nine of which had experience with tangible UIs. Despite the increased member of male participants, none of the tasks featured gender-influenced decisions. All participants successfully completed all three tasks. All participants successfully completed the first scenario with their first interaction attempt; the mean number of interactions/groups can be seen in Table 2. Four of the participants created groups; however this was unnecessary (average of 0.19 groups/participant) as the scenario did not require them. Sixteen participants completed the second scenario in their first interaction, four participants on their second interaction, with the remaining taking three interactions (average of 1.29 attempts/participant). Six participants falsely utilized groups in this scenario, believing groups were needed for interactions with more than two objects. For the final scenario 15 participants succeeded with their first interaction attempt, with five taking two attempts and one taking three. A mean of 2.38 groups were created per participant, with a goal of two groups per participant. We believe that the majority of users getting the desired outcome first time is important. It indicates that the participants' personal assumptions/expectations about how such a system should function, matched its functionality, both in terms of adapting their view of the problem to match the system and translating that knowledge to the system.

	Mean Interactions	**Mean Groups**
Task 1	1.00	0.19
Task 2	1.29	0.33
Task 3	1.46	2.38

Table 2: Average attempts per task

Participants reported that the system progressed in a similar logical order to how they would have described the scenario to another person, as well as how they thought through the problem mentally. One participant noted that it was easier to work through the scenario using the system instead of mentally, with another noting "I can't think of a more suitable way". One participant highlighted the flexibility offered by the system workflow as a benefit.

Participants reported in the questionnaire that introducing objects was both intuitive and easy to perform, with all participants responding favourably to intuitiveness and 19 responding favourably regarding ease of use. For creating groups, 19 participants gave a favourable rating regarding intuitiveness, with 16 giving favourable ratings regarding ease of group creation. For defining interaction rules, 18 participants gave favourable results for both intuitiveness and ease of creation. All participants gave favourable feedback saying that the system progressed in a similar way to how they thought the problem through both in their head, and followed how they would have explained it to another person.

The most problematic part of the study was the tracking systems, especially for skeleton tracking and gesture recognition, which varied greatly between users. Whilst the touch interaction was an improvement, accidental touches still occurred for all participants. Almost all participants directly addressed the tracking issues in their feedback. As such, participants did not feel the system to be overly responsive. Despite not being the focus of this work, we believe these problems will be addressed in the future. In spite of the tracking problems, users still found the system enjoyable to operate.

The vast majority of users found the guidance provided by the system to be almost ideal, with 20 participants giving favourable responses. The virtual keyboard had a nearly equal number of participants that liked and disliked it. Most users found the selected gestures both intuitive and appropriate for the tasks. Participants especially liked introducing and defining objects, as well as grouping objects and text-to-speech voice prompts by the system.

One interesting observation was how participants customized the examples, by naming the town, enemy and defensive army different names (e.g. NATO). This implied users were making cognitive connections between the objects and the context of the demonstration.

Overall, the results of the study demonstrated a favourable result for this form of ad-hoc interaction, even for participants without a technical background. All participants were able to complete the tasks after watching only a single interaction being defined. Results from the questionnaire reflect a strong approval for both TAM and SAR-TAM.

8 Applications of Generalized Tangible User Interfaces

As mentioned early in this paper, given UIs must be generalized since the designer does not know who the end user is or how it will be used, a major application of systems such as this would be to enable extensibility by (presumably) novice end-users. Their use as a means of externalizing internal thoughts into tangible artefacts leans towards their use in groupware to support computer-supported cooperative work and group collaboration. Such functionality would enable individuals to quickly externalize their internal views of how a system works, which instantly transfers to a collaborative interaction medium. As such, TAM has applications to tangible systems where end users are not the original developers. Additional functionality can be defined by the end user to address customization or new functions. In addition, core functionality currently programmed by the original system designers, could also be at least partially replaced and defined using the system.

Example systems for the application of these types of systems include real time planning (tangible war games table using live data from the field), education aids and any scenario where users could benefit from externalizing the problem into tangible problem solving.

9 Future Work

This paper describes our new TAM architecture and our first implementation in the SAR-TAM application. Arguably, the power of this system comes from the ability to map between properties/variables in a tangible manner, something we are yet to fully explore. We are investigating how to edit these mappings once created, which is an obvious limitation of the existing system. This would enable users to not only develop basic tangible systems, but when combined with virtual mappings, allow the user to develop complex applications. Example applications include support for education and problem solving as well as support for such small sub-systems including the previously highlighted multi-lingual virtual keyboards, tangible phone books, etc.

Key areas of future work include supporting:

- mapping for an object where it is both the input and the output (embodied TUIs),
- non-boolean events and temporal input and output,
- creation of physical-virtual tools (Marner et al., 2009),
- prerequisites for actions/rules ('daisy chaining'),
- undo/redo as well as saving/restoring saved configurations in a tangible realm, and
- debugging existing objects, rule sets and mapping configurations.

Following the results of the user study, we will also be exploring alternative control methods to supplement/replace the existing pose-based control system.

10 Conclusion

TAM enables the development of and interaction with novel TUIs with no background development. Through providing an abstracted set of interactions, novice users can program rules-based interactions, utilizing both individual objects and group-based interactions, offering type definition and substitution. Our system allows users to quickly externalize mental systems, as well as define novel TUIs. The user study results show TAM as both an effective and flexible means for allowing technical and non-technical users to create tangible systems ad-hoc. Future development will enable a wider range of interactions, which paired with a more advanced mapping system for programming logic, will allow for a richer set of interactions.

11 Acknowledgements

The authors would like to thank Thuong Hoang and Markus Broecker for proofreading the paper and the reviewers for their feedback and ideas.

12 References

BORCHERS, J., RINGEL, M., TYLER, J. & FOX, A. 2002. Interactive Workspaces: A Framework for Physical and Graphical User Interface Prototyping. *IEEE Wireless Communications,* 9**,** 64-69.

DEY, A. K., HAMID, R., BECKMANN, C., LI, I. & HSU, D. 2004. a CAPpella: programming by demonstration of context-aware applications. *Proc. of the SIGCHI conference on Human factors in computing systems.* Vienna, Austria: ACM.

FITZMAURICE, G. W. 1996. *Graspable User Interfaces.* Doctor of Philosophy, University of Toronto.

FITZMAURICE, G. W., ISHII, H. & BUXTON, W. A. S. 1995. Bricks: laying the foundations for graspable user interfaces. *Proc. of the SIGCHI conference on Human factors in computing systems.* Denver, Colorado, United States: ACM Press/Addison-Wesley Publishing Co.

GREENBERG, S. & FITCHETT, C. 2001. Phidgets: easy development of physical interfaces through physical widgets. *Proc. of the 14th annual ACM symposium on User interface software and technology.* Orlando, Florida: ACM.

HACKER, W. 1994. Action regulation theory and occupational psychology: Review of German empirical research since 1987. *German Journal of Psychology,* 18**,** 91-120.

HALBERT, D. C. 1984. *Programming by example.* Doctoral Dissertation, University of California.

HOLMAN, D. & VERTEGAAL, R. 2008. Organic user interfaces: designing computers in any way, shape, or form. *Communications of the ACM,* 51**,** 48-55.

ISHII, H. & ULLMER, B. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. *Proc. of the SIGCHI conference on Human factors in computing systems.* Atlanta, Georgia, United States: ACM.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A. & FITZGIBBON, A. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. *Proc. of the 24th annual ACM symposium on User interface software and technology.* Santa Barbara, California, USA: ACM.

KIRSH, D. & MAGLIO, P. 1994. On distinguishing epistemic from pragmatic action. *Cognitive Science,* 18**,** 513-549.

KJELDSEN, R., LEVAS, A. & PINHANEZ, C. 2003. Dynamically Reconfigurable Vision-Based User Interfaces. *In:* CROWLEY, J., PIATER, J., VINCZE, M. & PALETTA, L. (eds.) *Computer Vision Systems.* Springer Berlin/Heidelberg.

KLEMMER, S. R., LI, J., LIN, J. & LANDAY, J. A. 2004. Papier-Mache: toolkit support for tangible input. *Proc. of the SIGCHI conference on Human factors in computing systems.* Vienna, Austria: ACM.

MARNER, M. R., THOMAS, B. H. & SANDOR, C. 2009. Physical-virtual tools for spatial augmented reality user interfaces. *IEEE/ACM International Symposium on Mixed and Augmented Reality.* Orlando, Florida.

MARQUARDT, N., DIAZ-MARINO, R., BORING, S. & GREENBERG, S. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. *Proc. of the 24th annual ACM symposium on User interface software and technology.* Santa Barbara, California, USA: ACM.

MERRILL, D., KALANITHI, J. & MAES, P. 2007. Siftables: towards sensor network user interfaces. *Proc. 1st international conference on Tangible and embedded interaction.* Baton Rouge, Louisiana: ACM.

MYERS, B. A. 1986. Visual programming, programming by example, and program visualization: a taxonomy. *Proc. of the SIGCHI conference on Human factors in computing systems.* Boston, Massachusetts: ACM.

MYERS, B. A. 1992. Demonstrational interfaces: A step beyond direct manipulation. *Computer,* 25**,** 61-73.

REKIMOTO, J. & SAITOH, M. 1999. Augmented surfaces: a spatially continuous work space for hybrid computing environments. *Proc. of the SIGCHI conference on Human factors in computing systems.* Pittsburgh, Pennsylvania: ACM.

SCOTT, S. D., CARPENDALE, M. S. T. & HABELSKI, S. 2005. Storage bins: mobile storage for collaborative tabletop displays. *Computer Graphics and Applications, IEEE,* 25**,** 58-65.

SHARLIN, E., WATSON, B., KITAMURA, Y., KISHINO, F. & ITOH, Y. 2004. On tangible user interfaces, humans and spatiality. *Personal and Ubiquitous Computing,* 8**,** 338-346.

TRAVERS, M. 1994. Recursive interfaces for reactive objects. *Proc. of the SIGCHI conference on Human factors in computing systems.* Boston, Massachusetts: ACM.

ULLMER, B. & ISHII, H. 1997. The metaDESK: models and prototypes for tangible user interfaces. *Proc. of the 10th annual ACM symposium on User interface software and technology.* Banff, Alberta: ACM.

ULLMER, B. A. 2002. *Tangible interfaces for manipulating aggregates of digital information.* Doctor of Philosophy, Massachusetts Institute of Technology.

UNDERKOFFLER, J. & ISHII, H. 1999. Urp: a luminous-tangible workbench for urban planning and design. *Proc. of the SIGCHI conference on Human factors in computing systems: the CHI is the limit.* Pittsburgh, Pennsylvania: ACM.

VILLAR, N., BLOCK, F., MOLYNEAUX, D. & GELLERSEN, H. 2006. VoodooIO. *Proc. of ACM SIGGRAPH 2006 Emerging technologies* Boston, Massachusetts: ACM.

WALDNER, M., HAUBER, J., ZAUNER, J., HALLER, M. & BILLINGHURST, M. 2006. Tangible tiles: design and evaluation of a tangible user interface in a collaborative tabletop setup. *Proc. of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments.* Sydney, Australia: ACM.