Erschienen in: International Journal of Human-Computer Interaction ; 28 (2012), 11. - S. 737-747
https://dx.doi.org/10.1080/10447318.2012.715539

Design and Implementation of Post-WIMP Distributed User Interfaces with ZOIL

Hans-Christian Jetter, Michael Zöllner, Jens Gerken, and Harald Reiterer
University of Konstanz, Germany

"Interactive spaces" are physical environments or rooms for collaborative work that are augmented with ubiquitous computing technology. Their purpose is to enable a computer-supported collaboration between multiple users that is based on a seamless use of different devices for natural "post-WIMP" interaction (e.g., multitouch walls, interactive tabletops, tablet PCs, or digital pen and paper). However, to this day, there are no well-established guidelines or toolkits for designing and implementing such distributed user interfaces (DUIs). Therefore, this article introduces the *Zoomable Object-Oriented Information Landscape* (ZOIL), a novel design approach and software framework for post-WIMP DUIs in interactive spaces.

In the following, the *ZOIL design principles* are first introduced and illustrated. They provide recommendations and examples of DUI interaction design for interactive spaces. Then the different software patterns and architectures that have been employed for implementing the open-source *ZOIL software framework* are described. This framework facilitates the implementation of ZOIL's design principles in practice. Lessons learned from ZOIL's implementation are shared, and the implementation is discussed and compared with related work and approaches. The results of an evaluation of ZOIL with designers and developers conclude the article.

1. INTRODUCTION

Following Melchior, Grolaux, Vanderdonckt, and Van Roy's (2009) definition of a distributed user interface (DUI), a DUI is a user interface (UI) that has the ability to distribute some or all of its components across multiple monitors, devices, platforms, displays, and/or users (Melchior et al., 2009). This ability is essential in "interactive spaces" that augment physical environments such as design studios, meeting rooms or libraries with ubiquitous computing technology (Figure 1). In these interactive spaces, co-located users can collaboratively create, explore, search, and make sense of digital information using multiple different displays and devices (Haller et al., 2010; Jetter, Gerken, Zöllner, & Reiterer, 2010). Unlike previous DUIs that distribute components of graphical user interfaces (GUIs) (Bharat & Brown, 1994; Melchior et al., 2009) these interactive spaces diverge from the "**w**indow, **i**con, **m**enu, **p**ointing device" (WIMP) interaction style and use new input/output (I/O) modalities (e.g., tabletops with multitouch and tangibles, pen-operated tablets or walls, digital pen and paper). They thereby attempt to appear more natural and less obtrusive. Ideally this kind of post-WIMP interaction feels as familiar and fluid as using simple nondigital artifacts such as real-world pens, whiteboards, sheets of paper, or sticky notes within a natural social setting like a meeting or a brainstorming session.

Designing and implementing such post-WIMP UIs is particularly challenging. Although UI guidelines and toolkits for GUIs are well established, such standards do not yet exist for the post-WIMP world, for example, for tangible UIs (Shaer & Jacob, 2009). Distributed post-WIMP UIs further add to this challenge, because distribution and networking must be realized in addition to the already complex design and use of heterogeneous I/O technologies and event models.

In this article, we therefore introduce the *Zoomable Object-Oriented Information Landscape* (ZOIL), a novel approach for the design and implementation of post-WIMP DUIs for interactive spaces. ZOIL consists of *design principles* and a *software framework*: ZOIL's *design principles* are heuristics deduced from literature in human–computer interaction (HCI), Information Visualization (InfoVis), and Computer Supported Cooperative Work (CSCW). They provide recommendations and examples for appropriate conceptual models and DUI interaction design. The ZOIL open-source *software framework* for C# and Microsoft Windows Presentation Foundation (WPF) provides the necessary components for their implementation in own projects.

In the following, we first introduce and illustrate the ZOIL design principles. Then we describe the software patterns and our implementation of the ZOIL software framework. After sharing and discussing our lessons learned and comparing ZOIL with related work and approaches, we conclude

We thank Andreas Engl for his indispensable work on the initial design and implementation of the ZOIL software framework. We also thank the many colleagues and students from Konstanz who supported and evaluated ZOIL in their projects and courses.

Address correspondence to Hans-Christian Jetter, Human-Computer Interaction Group, Department of Computer and Information Science, Box D73, Universitaetsstrasse 10, University of Konstanz, 78457 Konstanz, Germany. E-mail: hans-christian.jetter@uni-konstanz.de

FIG. 1. Three examples for ZOIL-based interactive spaces. Left: AffinityTable (Geyer, Pfeil, Budzinski, Höchtl, & Reiterer, 2011). Center: Design Studio (Geyer & Reiterer, 2010). Right: Media Library (Demarmels, Huber, & Heilig, 2010) (color figure available online).

with the results of ZOIL's evaluation with designers and developers.

2. DESIGNING POST-WIMP DUIs FOR INTERACTIVE SPACES WITH ZOIL

The core of every ZOIL-based interactive space is a visual workspace called the "information landscape." It contains all kinds of objects from the application domain with their relevant metadata, content, mutual relations, and functions. This workspace is shared with all client devices (Figure 2) and can be accessed and directly manipulated following a consistent interaction model that exploits the devices' different form factors and I/O modalities (e.g., multitouch, tangibles, or gestures). This consistent model is based on a combination of an Object-Oriented User Interface (OOUI) with a Zoomable User Interface (ZUI). In the following, we describe this interaction model and ZOIL's UI architecture by introducing the six ZOIL design principles D1 to D6. Each principle focuses on an individual aspect of the interaction design and suggests appropriate design means and ends.

2.1. D1: Provide an OOUI for Direct Manipulation

In a ZOIL interactive space, it is not the applications but the objects of the application domain (e.g. documents, notes, sketches, images, videos) that become the first-level citizens of the UI. The necessary functions to work with these objects are directly attached to them. Functions are not scattered over multiple walled applications or other isolated silos of functionality. This notion of an OOUI instead of an application-oriented UI was first introduced to GUI design in the 1990s (Mandel, 1994; Roberts, Berry, Isensee, & Mullaly, 1998). Since then, only smaller parts of the original OOUI vision (e.g., the context menu) have become mainstream. The surrounding desktop metaphor remained application oriented, with many competing application metaphors and interaction styles (e.g., drag & drop of files and folders on the desktop, point & click of commands in wb applications, WYSIWYG editors for documents or spreadsheets).

OOUIs are based on the assumption that a more homogeneous and better UI design can be achieved by using the same principles that made object-oriented programming (OOP)

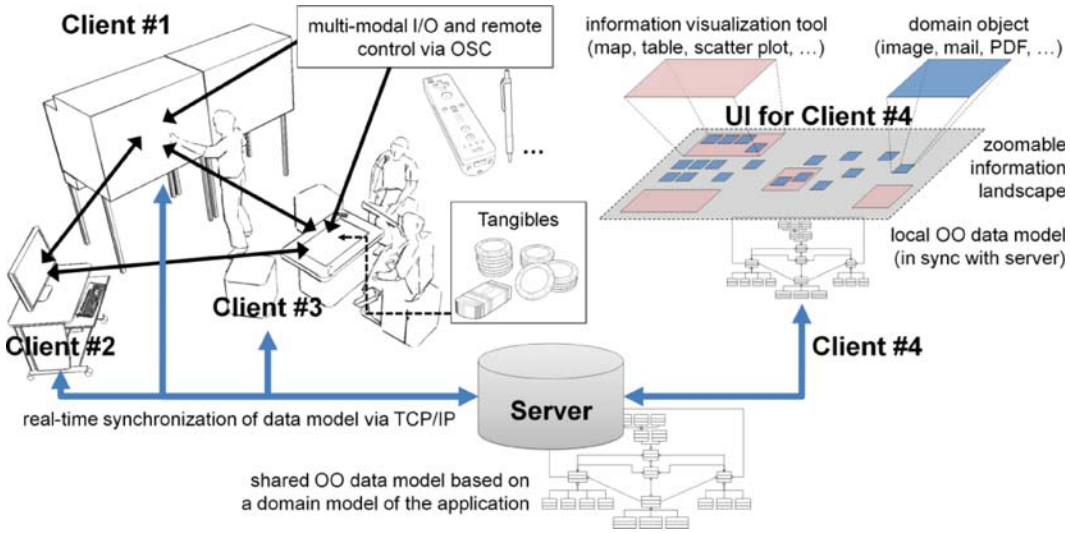

FIG. 2. In a ZOIL interactive space the data model of the workspace is distributed in realtime across multiple client devices and/or users. The right section shows how a ZOIL client visualizes the workspace's data model as a zoomable information landscape containing the domain objects (color figure available online).

remarkably successful. Both OOP and OOUIs share the hypothesis that object-orientation, that is, "thinking" about the world in terms of objects, their classes and functions, is a natural way of conceptualizing our environment and is close to the way we reason in our everyday world. Thus, an OOUI tries to represent the application domain as a world of objects of different classes with a concrete mapping between real-world entities and UI objects. An OOUI "focuses the users on objects—the 'things' people use to accomplish their work" (Roberts et al., 1998) instead of primarily focusing on the procedures people use to accomplish their work. To achieve this, all essential domain objects, their attributes, and attached functions are modeled in an object model, which is visually represented on the interface. The application domain itself becomes directly user accessible in a "model-world" and user tasks are carried out by continuous "direct manipulation" of objects inside of it (Hutchins, Hollan, & Norman, 1985).

To create OOUIs with a coherent and consistent conceptual model and to avoid exaggerated UI realism without expressive power (Jacob et al., 2008), OOUI approaches also bring OOP rigor to UI design: OO formalisms such as abstraction, inheritance, or polymorphism can be used to iteratively reduce the complexity of the UI's object model. This reduction increases coherence and consistency and makes it easier for users to apply logical reasoning or previous experiences while taking out novel and unknown tasks.

Working with an OOUI is different from working with an application-oriented UI where assumed work procedures or tasks are the first-level citizens of the UI and become manifest in the applications' sequences of pages, menus, dialogs, and widgets. In contrast, OOUIs have a "flexible structure-by object" instead of the "rigid structure-by function" of traditional GUIs (Mandel, 1994). This is particularly useful for collaborative multiuser tasks involving search or sensemaking where workflows are distributed and often volatile and ill-structured. Therefore, in previous work, we already argued that OOUIs are worth revisiting, and we demonstrated this with a case study of the model-based design of a visual information-seeking system (Jetter et al., 2010). A further reason why OOUIs are relevant for the post-WIMP world is that today's UI objects are not bound to screens anymore and can also become physical entities, tools, or tokens for tangible and/or virtual interaction (e.g., Figure 6). Here an OOUI's object model helps to systematically bridge the virtual and physical divide in post-WIMP interfaces: With the model it becomes easier to decide for or against a virtual, tangible, or hybrid representation of an object class by taking into account its associated properties, functions, and cardinalities.

2.2. D2: Provide a ZUI for Navigation With Semantic Zooming

The benefit of ZOIL's object-oriented interaction can be further enhanced by better exploiting our spatial memory and orientation when accessing and managing UI objects. In the traditional desktop metaphor, the functionality and content of the UI are scattered over multiple applications with ephemeral views, windows, or pages. Unlike the work environments of the real world, their spatial configuration is short-lived and continuously changing. The results are "mazelike" interfaces (Raskin, 2000) with a great cognitive load for navigating, finding information, or managing windows.

ZUIs have been introduced to facilitate user orientation and navigation by "tapping into our natural spatial and geographic ways of thinking" (Perlin & Fox, 1993, p. 57). This resonates with more recent post-WIMP design frameworks (Jacob et al., 2008) and guidelines (Wigdor & Wixon, 2011) that emphasize the power of the users' spatial cognition, environment skills, and environment awareness. Unlike the GUI's desktop, ZUIs are not limited to the visible screen size and resemble a canvas of infinite size and resolution for visual-spatial information management. All objects and their connected functionality can be accessed in an object-oriented manner without the use of file explorers and application windows simply by panning to the right spot in the information landscape and zooming in. Thus, ZUIs "can replace the browser, the desktop metaphor, and the traditional operating system. Applications per se disappear" (Raskin, 2000, p. 164).

ZOIL UIs employ ZUIs with "semantic zooming" (Perlin & Fox, 1993), which means that the growth of an object in display space is used not just to render the same object in a higher resolution but also to reveal different and more content and functionality (Figure 3). This smooth transition between iconic representation, metadata, and full-text/full-functionality

FIG. 3. Semantic zoom into a slide gradually reveals more controls and annotations (left). Zooming into a DVD reveals all metadata and a video stream with the actual movie (right) (color figure available online).

makes overlaying application windows or occluding renderings of details-on-demand unnecessary. This also resonates with design guidelines for natural UIs that recommend to focus UIs on their content and to use only a minimum of administrative controls or system states (Wigdor & Wixon, 2011).

2.3. D3: Provide Visualization Tools for Analytical Views

Natural spatial navigation in a ZUI is inherently limited regarding the amount and content of information objects. For example, repeated manual zooming and panning is not efficient for searching in the content of large numbers of objects or for getting an overview of the characteristics of an entire cluster. Furthermore, today's information objects carry extensive metadata like geo locations or user tags and ratings. For some tasks, faceted navigation of this metadata is far more efficient than manually browsing to the objects' respective virtual locations.

For this reason, a ZOIL UI also provides information visualization tools for creating alternative visual representations (e.g., maps, tables, plots, or bar charts; Figure 4). They can be instantiated as dynamic regions in the information landscape that aggregate metadata about great numbers of objects. Furthermore, these regions can provide controls for a dynamic filtering of objects (e.g., input fields for search queries or range sliders). As a consequence, such regions provide a functionality that is similar to today's virtual folders or desktop search engines. For greater flexibility, ZOIL's visualizations can be either integrated directly into the landscape (Figure 4 left) or used as free-floating "lenses" (Bederson et al., 1996) that provide alternative visual representations of the underlying objects (Figure 4 right).

2.4. D4: Provide Enough Space to Think and to Annotate

In today's computing practice, display space is not just used for navigating objects or presenting visualizations but also is a medium for supporting cognitively demanding tasks (Andrews, Endert, & North, 2010): Users use space to separate work zones (where items are used, read, or manipulated) from peripheral zones (where items are clustered or piled). They use space as external memory to minimize the amount of virtual navigation steps (e.g., less page flipping or switching between windows), and they let physical navigation augment memory. Also important are "incremental formalisms": Users gradually turn their initially random spatial configuration of items into increasingly formal and meaningful structures, for example, by sorting objects into different regions, clusters, piles, or even timelines.

We believe that ZUIs are a particular natural way of providing users with the necessary space to think without the cognitive overhead of overlapping windows or virtual 3D workspaces. ZOIL's zoomable information landscape provides infinite space that can be used freely, for example, to incrementally create arbitrary layouts, regions, or clusters of objects. Visual objects at different locations and scales can also serve as global or relative landmarks that support orientation in the landscape. Furthermore, the landscape is also a space for annotation: By default all regions of the landscape and objects can be visually annotated with finger or stylus input (Figure 3, left) or Anoto digital pens (Figure 5, right).

2.5. D5: Enable Distribution Across Different Devices

Today's great variety of interactive surfaces (e.g., pads, tablets, tabletops, wall displays) and input modalities (e.g., multitouch, stylus or Anoto pen) enables DUI users to choose the devices that are best suited for their current task or collaboration style. To distribute functionality across this range of devices, ZOIL uses a central server that shares the data model of the zoomable information landscape. For navigating or manipulating the landscape's content, users can use all I/O devices that can run a ZOIL client application to connect to this server. Thereby, all the changes that are made to the content of the

FIG. 4. Left: A ZOIL user interface with a map visualization for photo browsing (Jetter, Engl, Schubert, & Reiterer, 2008). Different ZOIL visualizations for a physical lens on a tabletop (right) (color figure available online).

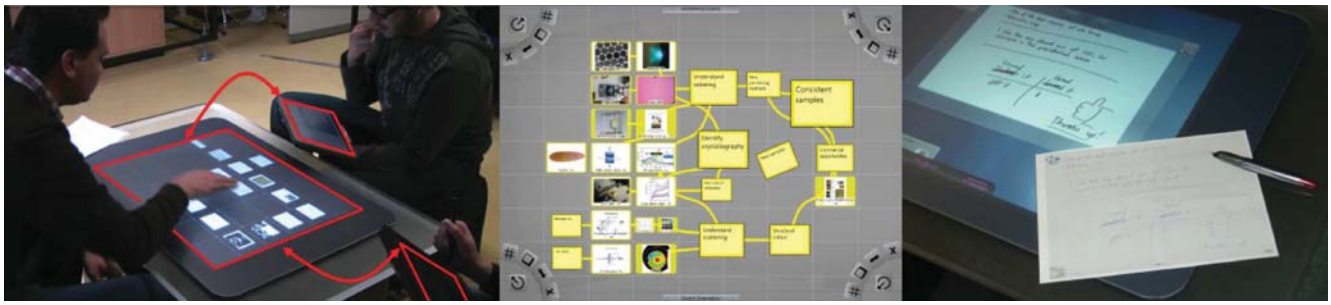

FIG. 5. Sensemaking of scientific content in DeskPiles (Project DeskPiles, 2010) with tabletop and tablets (left & center). Anoto digital pen & paper for annotation (right) (color figure available online).

landscape with a client device are synchronized with all other connected clients via the server in real time (Figure 2).

By sharing the underlying data model instead of the view or the displayed content on the "pixel-level," each client device can render its own view of the shared landscape using its native device-specific resolution and its I/O capabilities and modalities. A connected client is also free to render an arbitrary section of the landscape in own visualization styles (e.g., for a user-, task-, or device-specific representation) so that different users can work in the same workspace at the same time but at different locations and using different visual representations. Other scenarios involve different modes of coupling between devices, for example, a tabletop can serve as an overview display for a wall-sized high-resolution display. In this case, the physical lens on the tabletop from Figure 4 can be used to control what part of the landscape becomes visible on the wall. In conclusion, sharing the data model enables "homogeneous" and "heterogeneous" DUIs (Bharat & Brown, 1994), that is, either the DUI can offer multiple instances of the same user interface at different sites or the functionality provided at each site may be different.

2.6. D6: Enable and Support Multiuser Collaboration

ZOIL's aforementioned client-server architecture provides the infrastructure for co-located collaboration by making the visual workspace accessible from multiple different devices. However, the UIs of the individual devices have to be designed carefully to truly support collaboration—in particular when UIs are shared by multiple users (e.g., on large displays or tabletops). Based on the experiences from the design and evaluation of the ZOIL-based tabletop system Facet-Streams for collaborative search (Jetter, Gerken, Zöllner, Reiterer, & Milic-Frayling, 2011), we identify three design goals for ZOIL multiuser UIs:

1. A visual and/or tangible externalization of the work process must be provided to create the necessary awareness for the group's tasks, current state of work, and the next steps. The spatial arrangement of objects, object piles, visualizations, and annotations in ZOIL's information landscape can serve this purpose (Figure 5, center). In Facet-Streams, the externalization of the search process is provided above the information landscape with tangible filter tokens that are connected by a visual filter/flow metaphor (Figure 6).
2. The externalization should support transitions between phases of loosely coupled parallel work and tightly coupled collaboration. ZOIL's information landscape is a natural medium for this purpose: In the zoomable information landscape, users can use different regions in space and scale on different devices to smoothly switch between personal and shared spaces or work zones. The spatial and geographic nature of the workspace helps to maintain the awareness of how the current personal view relates to the entire workspace

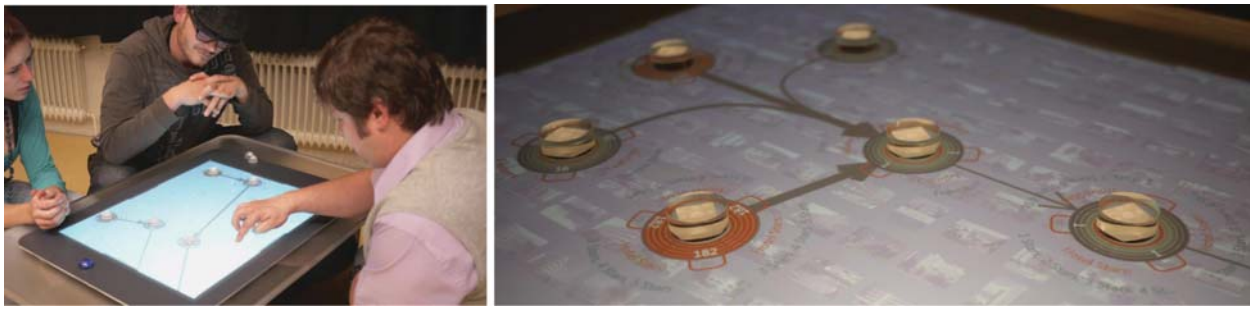

FIG. 6. The ZOIL-based Facet-Streams system for collaborative search uses tangibles and radial controls and labels (Jetter, Gerken, Zöllner, Reiterer, & Milic-Frayling, 2011) (color figure available online).

and the group's task. Similarly, the tangible filter/flow network of Facet-Streams is an externalization of the current state of the group search task that can be easily split into personal subnetworks, which then can be individually revised and later merged to the group network again.

3. For co-located collaboration, the hardware and UI must support and process parallel interactions of multiple users with concurrent input at multiple points (e.g., a tabletop with true multitouch and tangible interaction). To ensure effective around-the-table collaboration, also the reachability and readability of UI elements from all sides must be considered (e.g., by using radial labels and controls). Such a design for equitable face-to-face collaboration helps users to better employ their familiar "social skills and awareness" (e.g., verbal and nonverbal communication, social protocols) to coordinate tasks and working styles (Jacob et al., 2008).

3. IMPLEMENTING POST-WIMP INTERACTIVE SPACES WITH THE ZOIL SOFTWARE FRAMEWORK

The design principles D1 to D6 introduce the defining features of a ZOIL interactive space from a designer's or user's perspective. For programmers, their actual technical realization is particularly challenging. For this reason, we have created the ZOIL software framework to facilitate ZOIL's implementation. The framework is an extensible collection of components and classes for C# and WPF with approximately 4.000 lines of code. Figure 7 gives an overview of the functionality of the framework and the components it is built on.

As illustrated, the ZOIL framework serves as a kind of middleware that provides high-level functionality in the areas of presentation and interaction, network communication, and persistence and synchronization. These key aspects of any ZOIL design can be realized using the framework without the need to directly access the underlying lower level libraries or APIs. We share the ZOIL framework as open-source under the BSD License (ZOIL Project, 2010). In the light of the framework's volume and scope, we focus our description here on selected features and report about our design decisions by formulating software patterns P1 to P4.

3.1. P1: Client–Server Architecture for Transparent Persistence and Synchronization

For realizing multidevice scenarios (D5), ZOIL employs a central database server for providing immediate persistence and real-time synchronization to all client applications and devices. All objects in ZOIL's information landscape are stored on this server and thus keep their state, properties, or location even when all clients are closed or unexpectedly shut down. Furthermore, all changes made to an object on one client are synchronized via the server with all other connected clients in real time using a nonlocking optimistic synchronization scheme.

To achieve this, we created two components using the C# API of the embedded object database db4o (Versant Corp.; http://www.db4o.com): a ZOIL data backend that is part of all ZOIL client applications and a ZOIL server application that provides ACID properties for all transactions and a GUI for administration. ZOIL's client-side data backend uses db4o's support for Transparent Persistence (TP). This enables programmers to access, change, and synchronize persistent data with the same ease as nonpersistent local data models in main memory. Programmers can simply change the states and locations of objects on the client side using standard C# code. Transparent to the programmer, the TP implementation in ZOIL's backend observes and collects all these changes and transmits them via TCP/IP to the server in regular intervals (typically 100 ms). The server then informs all other clients about the changes, so that the clients' backends can retrieve and execute them within 100 to 300 ms (in typical setups).

In addition to the TP feature, db4o also helps to facilitate iterative design due to its object-oriented nature: Unlike with relational databases, code changes in object definitions are automatically detected using reflection during run-time without the need for manually updating any object-relational mappings or table schemata.

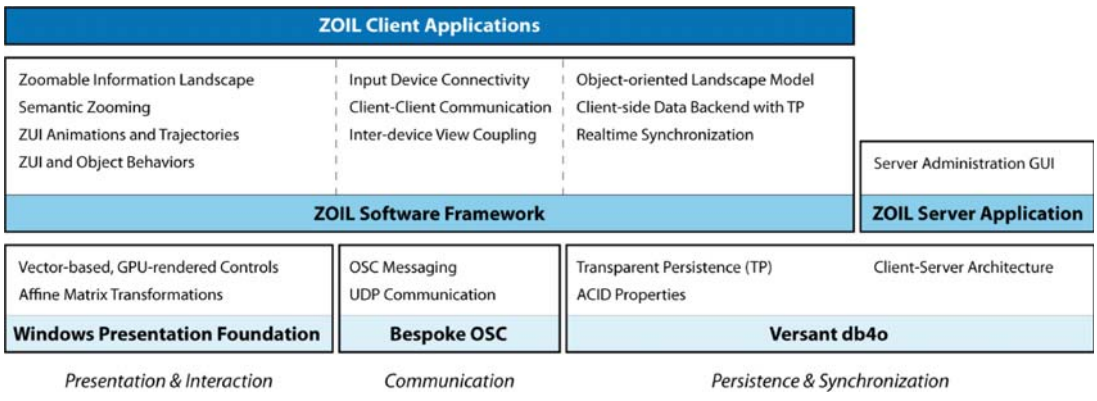

FIG. 7. The ZOIL software framework sits between the individual ZOIL client applications at the top and lower level APIs at the bottom. *Note.* It also provides a ZOIL server application with GUI (color figure available online).

3.2. P2: Declarative Definition of Zoomable UI Components

A defining feature of ZOIL is its comprehensive support for authoring and implementing rich ZUIs with semantic zooming using WPF's declarative XML-based Extensible Application Markup Language (XAML). Following XAML's declarative paradigm, we have extended the original set of XAML elements and WPF's native controls with ZOIL-specific components for defining semantic zooming and other behaviors without the need for any procedural C# code (Figure 8, left). The different appearances of objects at different zoom levels are selected by our ZComponentFrames container control. It selects an appearance based on the available screen estate for the control and uses an opacity animation to smoothly switch between them during zooming.

Unlike Java SWT, Java Swing, or WinForms, WPF natively supports vector-based and hardware-accelerated rendering of controls. This enables ZOIL designers to directly use the full range of WPF controls (e.g., sliders, player controls) in ZUIs without pixelation or the need for wrapper classes as in Piccolo (Bederson, Grosjean, & Meyer, 2004). This also allows programmers to easily integrate media content such as video streams or 3D models and increases the prototyping efficiency in comparison to other hardware-accelerated graphics APIs with only rudimentary sets of user controls (e.g., OpenGL, DirectX, XNA).

3.3. P3: Declarative Definition of Behaviors With the Attached Behavior Pattern (ABP)

For designing object-oriented interaction it is not sufficient to define an object's visual appearance alone. Objects also have to be assigned the desired behaviors: Can an object be dragged, resized, or rotated using multitouch manipulations? Does an object simulate physical behavior such as inertia or friction during these manipulations? Is the object a zoom target, so that a click or tap on it starts a zooming animation into the object until it covers the screen?

To help designers to easily assign such behaviors, our framework makes extensive use of the ABP (Gossman, 2008). ABP encapsulates behaviors into a class outside the object's class hierarchy and allows it to be applied on classes or individual objects. When using WPF this threshold can be further lowered using XAML instead of C# (Figure 8, right). Thus, by using ABP, the ZOIL framework can provide designers with an extensible library of useful off-the-shelf behaviors that can be easily assigned to objects. We believe that ABP introduces a remarkably simple way of defining interactive behavior that facilitates iterative design without the need for procedural C# code or class hierarchies.

3.4. P4: Input Device and Client-Client Communication With Open Sound Control (OSC)

The ZOIL framework provides simple ways to connect to popular frameworks, platforms, or libraries for multimodal and tangible interaction such as TUIO, Squidy (König, Rädle, & Reiterer, 2010) or OpenInterface (Lawson, Al-Akkad, Vanderdonckt, & Macq, 2009). Applications can use ZOIL's support for the stateless UDP-based protocol OSC that has now emerged as a de facto standard for transporting post-WIMP I/O events. OSC also allows to tightly couple two or more clients' views as described in D5: By sending or broadcasting OSC commands between clients it is possible to control the viewport parameters of a remote client from a local client. This enables scenarios where the parameters of the currently displayed section of the landscape are continuously transferred from a local device to a remote device or vice versa. For example, this can be used to implement a physical "overview+detail" lens on a tabletop that controls a wall display (similar to Figure 4).

4. COMPARISON OF ZOIL WITH PREVIOUS AND RELATED WORK

After introducing the ZOIL design principles and software framework, we can compare ZOIL with previous and related work. For space reasons, we focus our comparison here only on the most essential work that shares our goals or approaches. A more comprehensive overview of publications on multiuser and multidisplay designs and systems is provided by Terrenghi, Quigley, and Dix (2009).

The main source of inspiration for ZOIL was Streitz et al.'s (1999) pioneering vision of i-Land and its "roomware." They

```
<ZEmail x:Class="ZComponent" … >
 <ZComponentFrames>
  <ZComponentFrame WidthNeeded="0">
   … appearance at 1st zoom level …
  </ZComponentFrame>
  <ZComponentFrame WidthNeeded="150">
   … appearance at 2nd zoom level …
  </ZComponentFrame>
  … add appearances for more zoom levels here …
 </ZComponentFrames>
</ZEmail>
```

```
<ZPhoto
    ZObjectDragDropBehavior.IsDraggable="True"
    ZObjectResizeBehavior.IsResizable="False"
    ZObjectRotateBehavior.IsRotatable="True"
    ZObjectPhysicsBehavior.HasInertia="True"
    ZInformationLandscape.ZoomTarget="True">
    ZInformationLandscape.ZoomMargin="5,10,5,5">
    <!-- further definition of ZPhoto -->
</ZPhoto>
```

FIG. 8. XAML code for defining the semantic zoom of an e-mail object (left). XAML for making ZPhoto a draggable, resizable, and rotatable zoom target (right) (color figure available online).

introduced example scenarios, roomware devices (e.g., interactive walls, chairs, and tabletops), and a software framework in Smalltalk. However, the UI design and the framework were not discussed in greater detail, and no studies with users or developers were conducted to evaluate them. Similar to ZOIL, Streitz et al.'s framework already provided a shared-object space across all devices and a virtual location metaphor for structuring collaboration. Although i-Land used a metaphor of discrete "physical rooms," ZOIL uses locations in the zoomable information landscape for smooth navigation and transitions between collaborative tasks. This concept of a zoomable landscape for ZOIL was originally suggested in the context of single-user information management and an early proof-of-concept without any UI distribution was demonstrated in (Jetter, Engl, Schubert, & Reiterer, 2008).

Similar to the vision of i-Land, the iRoom and its iROS meta-operating system enabled collaborative use of one or more large displays with portable devices in a local physical space (Johanson, Fox, & Winograd, 2002). Key contributions of iRoom were new GUI widgets and interaction techniques for fluid interaction (e.g., FlowMenu). However, alternatives to the GUI's traditional desktop or file system were not part of this research. Similar to ZOIL, iROS aimed at a minimal barrier-to-entry for developers. However, ZOIL or Shared Substance (see next) provide a much closer connection between data and events than iROS, which results in a conceptually simpler and more coherent approach for the developer (Gjerlufsen, Klokmose, Eagan, Pillias, & Beaudouin-Lafon, 2011).

Hugin is a Java-based InfoVis framework that, similar to ZOIL, provides visualizations for real-time co-located and remote collaboration using multiple tabletops (Kim, Javed, Williams, Elmqvist, & Irani, 2010). Hugin's visualizations and access control are more advanced than those integrated in ZOIL. However, the size of Hugin's shared workspace is currently restricted to the physical size of the screen. Thus, unlike in ZOIL, zooming and panning cannot be used to provide more virtual space for sensemaking. Also, moving in space and scale cannot be used for coordinating collaboration or to establish task- or group-specific regions. Like Hugin, ZOIL has already been used for tabletops (Jetter et al., 2011) and to integrate them into multidisplay interactive spaces (Demarmels, Huber, & Heilig, 2010; Geyer, Pfeil, Budzinski, Höchtl, & Reiterer, 2011; Geyer & Reiterer, 2010; sProject DeskPiles, 2010). However, all this previous work focused on the individual applications and, unlike this article, it did not introduce ZOIL as a design approach or software framework.

Shared Substance is a recent sophisticated data-oriented framework and distributed application model for multisurface environments (Gjerlufsen et al., 2011). Each client application or device uses the nodes of a hierarchical scene graph to store or access the application state, system resources or input device streams. Unlike in ZOIL, these nodes can be selectively shared over a network with other clients using discovery mechanisms without a central server. ZOIL does not provide selective sharing or discovery mechanisms and focuses only on synchronization and persistence of the entire workspace. Shared Substance also enables a much more selective replication and "mounting" of nodes using remote method invocation. However, persistence has to be implemented by the application developers, and Shared Substance is also not aimed at formulating generic design principles.

5. EVALUATION OF THE ZOIL SOFTWARE FRAMEWORK

For evaluating user interface toolkits, Myers, Hudson, and Pausch (2000) identified two key qualities: the *threshold* ("how difficult it is to learn how to use the tool?") and the *ceiling* ("how much can be done using the tool?"). The ideal toolkit has a low threshold and a high ceiling. To empirically evaluate the threshold and ceiling of ZOIL in practice, we conducted two user studies with student designers and developers using an approach for API usability evaluation based on concept maps (Gerken, Jetter, Zöllner, Mader, & Reiterer, 2011).

In the first study, we observed eight participants (graduate-level students of computer science) working in teams of two on individual self-chosen projects. The study lasted 4 weeks beginning with an introductory 2-hr lecture about the ZOIL framework. After that, each of the four groups worked independently. During weekly interview sessions each group created a concept map to give us a better understanding of the participants' mental model of the ZOIL framework, and we could analyze each group's progress and the problems they encountered. In a second study, 11 participants (nine graduate-level and two undergraduate students of computer science) were working in four teams of two and one team of three. Each team was given the same task with identical requirements for designing and implementing a ZOIL-based user interface for search and sensemaking of hotel data and hotel photos (Figure 9, bottom). The study lasted 5 weeks starting with an initial briefing session that was followed by weekly 1-hr interview sessions per group. Some results of this study that were related to model-based design of OOUIs were previously published in Jetter et al. (2010).

As reflected in the study designs, our primary goal was to collect rich, qualitative data from realistic projects instead of measuring quantitative data (e.g., task times or task completion rates) in artificial controlled experiments. However, we also collected quantitative subjective user ratings with questionnaires to gather enough data for the discussion of ZOIL's ceiling and threshold next.

5.1. ZOIL's Ceiling

Concerning the ceiling of the ZOIL framework, Figure 1 and Figure 9 provide a visual impression of the great variety and high fidelity of the interactive prototypes that were created, demonstrated, and used during and after the studies. VizDash (Study 1) used a tabletop and two large, high-resolution displays for the collaborative generation, exploration, and presentation

FIG. 9. VizDash uses a tabletop to generate charts for discussion on a wall display (top). HotelBrowser filters and zooms into geo-referential hotel data and content (bottom) (color figure available online).

of charts from a business database. Later, similar concepts were realized with ZOIL and published for collaborative affinity diagramming (Geyer et al., 2011). HotelBrowser (Study 2) enabled users to filter and semantically zoom into geo-referential hotel data and user-generated content (Figure 9). A further prototype from Study 1 enabled playful collaborative search in the IMDb movie database with a tabletop and a large display. This inspired the ZOIL-based search tools SearchToken from Figure 1 (right; Heilig et al., 2011) and Facet-Streams from Figure 6 (Jetter et al., 2011). A further project from Study 1 provided a zoomable landscape on the Windows desktop for spatially managing activities and files. Files could be dragged or pasted into the landscape using the Windows clipboard. Furthermore, visual links could be established between items to group them and navigate between them. This project inspired what later became DeskPiles, a ZOIL-based tool for managing, annotating, and sensemaking of scientific content with tablets and tabletops (Figure 5, left & center).

In conclusion, the systems created during and after the study reveal the high ceiling and diversity of what can be achieved when using the ZOIL framework. The framework benefits from its use of C#, .NET, and WPF because of its interoperability with most APIs and SDKs (e.g., Microsoft Surface SDK) for the Windows operating system. However, the questionnaires from the participants of the study also hint at room for improvement: At the end of the first study, on a scale from 1 (*very low ceiling*) to 7 (*very high ceiling*), all four participating groups have assessed ZOIL's ceiling with 5. At the end of the second study, on a scale from 1 (*very low ceiling*) to 5 (*very high ceiling*), the mean rating of the 11 participants was 3.6 ($SD = 0.5$). Because these ratings are only slightly above neutral, we analyzed the qualitative data collected during the interviews and found two repeatedly criticized aspects of ZOIL.

First, because ZOIL is based on WPF, it also inherits a critical weakness of this platform: Currently there is no native support for rendering rotated or scaled Web, PDF, or MS Office content in WPF. This is a great disadvantage when trying to realize around-the-table interaction with real-world content. Second, participants struggled with the support of drag-and-drop in the ZOIL framework. Although the "IsDraggable" behavior (Figure 8) is a very simple way to make objects movable and to enable changing locations, participants also asked for an equivalent support of drag-and-drop for initiating actions, for example, after dropping an object onto another. Although both problems could have been solved by using third-party controls or implementing own behaviors, the effort for this was apparently perceived as too great and thus negatively influenced ZOIL's perceived ceiling during the evaluation.

5.2. ZOIL's Threshold

Due to its nature as a software framework that builds upon a complex API such as WPF, ZOIL cannot be considered a low-threshold toolkit or editor for rapid prototyping. All studied projects lasted 4 or 5 weeks and they involved development effort in C# and XAML with the lines of code varying between 700 and 2.800. To get a better estimate for the necessary development effort, we asked the participants of Study 2 to self-report

the hours they spent on the project. The result was a mean of 15 hr per week and per participant, so one third to one half of a full-time developer. Although this clearly indicates a high threshold, it is still encouraging regarding the fact that half of the participants in Study 1 and all participants in Study 2 had no prior experience with C#, WPF, and XAML at all. Based on these observations, we conclude that ZOIL's threshold is still low enough that a two-person team of C# and WPF novices is able to create a complex distributed post-WIMP UI with ZOIL in a 1- to 3-week full-time project. This resonates with the slightly below neutral assessments of ZOIL's threshold by the four groups of Study 1 with a mean score of 3.25 ($SD = 0.96$) on a scale from 1 (*easy to learn*) to 7 (*difficult to learn*) and the 11 C# and WPF novices of Study 2 with a mean score of 2.6 ($SD = 0.92$) on a scale from 1 (*easy to learn*) to 5 (*difficult to learn*).

5.3. Other Findings

The studies also revealed a reoccurring theme across all groups relevant for post-WIMP DUI programming in general. All participants strongly approved of the ABP described in P3. Although the underlying mechanisms were mostly not understood in detail, the simple way of selecting and assigning behaviors to objects was considered as "logical" or "natural." On the contrary, groups in both studies criticized ZOIL's Model-View-Controller-style pattern for domain objects as overly complicated, unnecessary, and artificial compared to the naturalness of ABP. We believe that this hints at a more general theme that is analogous to the common desire for concrete mappings and monolithic toolkits in OOP (Bederson et al., 2004). Similar to the user of a post-WIMP UI who perceives an UI object as a single physical entity (e.g., a virtual or tangible object on a tabletop with size, mass, friction), post-WIMP programmers would like to create such objects and define their behaviors and properties without creating many artificial software objects that have no counterparts in the real world. Although ABP supports this kind of concrete mapping and natural programming, common model-view separation patterns do the exact opposite and disintegrate a UI object into three components from which only the view has a real-world counterpart and physical dimensions. Although participants accepted the necessity for a separation of view and model, they still would strongly prefer more natural mappings between code objects and real-world objects and more declarative approaches for defining properties and behaviors.

6. CONCLUSION

In this article we have introduced the ZOIL approach for creating DUIs for post-WIMP interactive spaces. We have formulated the ZOIL design principles D1 to D6 based on existing literature from HCI, InfoVis, and CSCW and how they were applied in different examples of ZOIL-based systems taken from recent publications. We have also described the different software patterns and architectures P1 to P4 that we have employed for implementing the open-source ZOIL software framework. A discussion of previous and related work has compared our ZOIL approach to related design and implementation frameworks for post-WIMP DUIs in interactive spaces. Finally, our evaluation of ZOIL's threshold and ceiling with designers and developers revealed ZOIL's generally high ceiling with some room for improvement concerning the support of different content types and drag-and-drop. ZOIL's threshold turned out to be higher than that of low-threshold visual UI toolkits but still low enough to create complex DUIs without prior knowledge of C# or WPF in 1 to 3 weeks. We consider the results as encouraging and believe that further reduction of the frameworks complexity, for example, more use of the ABP and less polylithic Model-View-Controller-style mappings, can further improve ZOIL's threshold and by this also the efficiency of designing and implementing post-WIMP DUIs for multiuser and multidevice spaces in general.

REFERENCES

Andrews, C., Endert, A., & North, C. (2010). Space to think: Large high-resolution displays for sensemaking. *CHI '10*, 55–64.

Bederson, B., Grosjean, J., & Meyer, J. (2004). Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, *30*, 535–546.

Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., & Furnas, G. W. (1996). Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages and Computing*, *7*, 3–31.

Bharat, K., & Brown, M. H. (1994). Building distributed, multi-user applications by direct manipulation. *UIST '94*, 71–80.

Demarmels, M., Huber, S., & Heilig, M. (2010). Meet me in the Library! *SIDeR '10*, 61–66.

Gerken, J., Jetter, H.-C., Zöllner, M., Mader, M., & Reiterer, H. (2011). The concept maps method as a tool to evaluate the usability of APIs. *CHI '11*, 3373–3382.

Geyer, F., Pfeil, U., Budzinski, J., Höchtl, A., & Reiterer, H. (2011). AffinityTable—A hybrid surface for supporting affinity diagramming. *INTERACT '11*, 477–484.

Geyer, F., & Reiterer, H. (2010). A cross-device spatial workspace supporting artifact-mediated collaboration in interaction design. *CHI EA '10*, 3787–3792.

Gjerlufsen, T., Klokmose, C. N., Eagan, J., Pillias, C., & Beaudouin-Lafon, M. (2011). Shared substance: Developing flexible multi-surface applications. *CHI '11*, 3383–3392.

Gossman, J. (2005). Introduction to Model/View/ViewModel pattern for building WPF apps [Web log post]. Retrieved from http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx

Gossman, J. (2008). The Attached Behavior pattern [Web log post]. Retrieved from http://blogs.msdn.com/b/johngossman/archive/2008/05/07/the-attached-behavior-pattern.aspx

Haller, M., Leitner, J., Seifried, T., Wallace, J. R., Scott, S. D., Richter, C., . . . Hunter, S. (2010). The NiCE discussion room: Integrating paper and digital media to support co-located group meetings. *CHI '10*, 609–618.

Heilig, M., Huber, S., Gerken, J., Demarmels, M., Allmendinger, K., & Reiterer, H. (2011). Hidden details of negotiation: The mechanics of reality-based collaboration in information seeking. *INTERACT 2011*, pp. 622–639.

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1985). Direct manipulation interfaces. *Human–Computer Interaction*, *1*, 311–338.

Jacob, R. J., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., & Zigelbaum, J. (2008). Reality-based interaction: A framework for post-WIMP interfaces. *CHI '08*, 201–210.

Jetter, H.-C., Engl, A., Schubert, S., & Reiterer, H. (2008). Zooming not zapping: Demonstrating the ZOIL user interface paradigm for ITV applications. *Adjunct Proceedings of European Interactive TV Conference (EUROITV '08)*, 239–240.

Jetter, H.-C., Gerken, J., Zöllner, M., & Reiterer, H. (2010). Model-based design and prototyping of interactive spaces for information interaction. *HCSE '10*, 22–37.

Jetter, H.-C., Gerken, J., Zöllner, M., Reiterer, H., & Milic-Frayling, N. (2011). Materializing the query with facet-streams: a hybrid surface for collaborative search on tabletops. *CHI '11*, 3013–3022.

Johanson, B., Fox, A., & Winograd, T. (2002). The Interactive Workspaces Project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, *1*, 67–74.

Kim, K., Javed, W., Williams, C., Elmqvist, N., & Irani, P. (2010). Hugin: A framework for awareness and coordination in mixed-presence collaborative information visualization. *ITS '10*, 231–240.

König, W., Rädle, R., & Reiterer, H. (2010). Interactive design of multimodal user interfaces—Reducing technical and visual complexity. *Journal on Multimodal User Interfaces*, *3*, 197–213.

Lawson, J., Al-Akkad, A., Vanderdonckt, J., & Macq, B. (2009). An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. *EICS 09*, 245–254.

Mandel, T. (1994). *The GUI-OOUI War, Windows vs. OS/2: The designer's guide to human-computer interfaces*. New York, NY: Van Nostrand Reinhold.

Melchior, J., Grolaux, D., Vanderdonckt, J., & Van Roy, P. (2009). A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. *EICS 2009*, 69–78.

Myers, B., Hudson, S. E., & Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions of Computer–Human Interactions*, *7*, 3–28.

Perlin, K., & Fox, D. (1993). Pad: An alternative approach to the computer interface. *SIGGRAPH '93*, 57–64.

Project DeskPiles. (2010). *DeskPiles—Microsoft research*. Retrieved from http://research.microsoft.com/en-us/projects/deskpiles/default.aspx

Raskin, J. (2000). *The humane interface: New directions for designing interactive systems*. New York, NY: ACM Press/Addison-Wesley.

Roberts, D., Berry, D., Isensee, S., & Mullaly, J. (1998). *Designing for the user with OVID: Bridging user interface design and software engineering*. Indianapolis, In: Macmillan Technical.

Shaer, O., & Jacob, R. J. (2009, November). A specification paradigm for the design and implementation of tangible user interfaces. *TOCHI*, *16*(4), 20:1–20:39.

Streitz, N. A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., . . . Steinmetz, R. (1999). i-LAND: An interactive landscape for creativity and innovation. *CHI '99*, 120–127.

Terrenghi, L., Quigley, A., & Dix, A. (2009). A taxonomy for and analysis of multi-person-display ecosystems. *Personal and Ubiquitous Computing*, *13*, 583–598.

Wigdor, D., & Wixon, D. (2011). *Brave NUI World : Designing natural user interfaces for touch and gesture*. New York, NY: Morgan Kaufman.

ZOIL Project. (2010). *ZOIL framework*. Retrieved from http://zoil.codeplex.com

ABOUT THE AUTHORS

Hans-Christian Jetter is a research assistant and PhD candidate in the HCI group of the University of Konstanz. His research focuses on the design and implementation of post-WIMP interaction and information visualization techniques for collaborative, multi-touch and tangible user interfaces. He is also interested in cognitive models for embodied interaction.

Michael Zöllner has recently received a M.Sc. in Information Engineering at the University of Konstanz. He is interested in software engineering and architectural patterns for post-WIMP user interfaces. For this purpose, he uses .NET, C# and Windows Presentation Foundation and is the main contributor to the ZOIL software framework.

Jens Gerken has received a PhD in the HCI group of the University of Konstanz in 2011. His research focused on longitudinal research methods in HCI and usability evaluation. He is now working as a senior consultant for interactive technologies for the ICT Innovative Communication Technologies AG in Kohlberg, Germany.

Harald Reiterer is full professor for Human-Computer Interaction in the Department of Computer & Information Science at the University of Konstanz in Germany. He established this HCI lab in 1997 with an initial focus on visual information seeking systems. In recent years his research has shifted toward more holistic interaction paradigms for the post-WIMP world.