VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces

Vildan Tanriverdi and Robert J.K. Jacob
Department of Electrical Engineering and Computer Science
Tufts University
Medford, MA 02155
{vildan | jacob} @eecs.tufts.edu

ABSTRACT

Compared to conventional interfaces, Virtual reality (VR) interfaces contain a richer variety and more complex types of objects, behaviors, interactions and communications. Therefore, designers of VR interfaces face significant conceptual and methodological challenges in: a) thinking comprehensively about the overall design of the VR interface; b) decomposing the design task into smaller, conceptually distinct, and easier tasks; and c) communicating the structure of the design to software developers. To help designers to deal with these challenges, we propose a Virtual Reality Interface Design (VRID) Model, and an associated VRID methodology.

Categories and Subject Descriptors

Realism-Virtual reality, Computing Methodologies

General Terms

Design, Theory

Keywords:

Virtual reality, user interface software, design model, design methodology

1. INTRODUCTION

Virtual Reality (VR) is seen as a promising platform for development of new applications in many domains such as medicine, entertainment, science and business [1, 17, 29, 31]. Despite their potential advantages, however, we do not yet see widespread development and use of VR applications in practice. The lack of proliferation of VR applications can be attributed partly to the challenges of building VR applications [2, 8]. In particular, interfaces of VR applications are more complex and challenging to design compared to interfaces of conventional desktop based applications [7, 9]. VR interfaces exhibit distinctive visual, behavioral and interaction characteristics.

Visual characteristics. While conventional interfaces mainly use 2D graphical displays, VR interfaces use both 2D and 3D displays. A major goal of virtual environments is to provide users with realistic environments. In order to provide the sense of "being there," VR interfaces heavily use 3D graphical displays. Design and implementation of 3D graphical displays are usually more difficult than 2D displays.

Conventional interfaces typically contain only virtual, computer-generated objects. VR interfaces, on the other hand, may contain both virtual and physical objects that coexist and exchange information with each other, as in the case of augmented reality systems. The need to properly align virtual and physical objects constitutes an additional challenge in VR interface design [3].

Behavioral characteristics. In conventional interfaces, objects usually exhibit passive behaviors. In general, they have predetermined behaviors that are activated in response to user actions. Therefore, communication patterns among objects are usually deterministic. VR interfaces contain both real world-like objects and magical objects that exhibit autonomous behaviors. Unlike passive objects, autonomous objects can change their own states. They can communicate with each other and affect each other's behaviors and communication patterns. Therefore, designing object behaviors is more challenging in VR interfaces.

Interaction characteristics. While conventional interfaces support mainly explicit style interactions, VR interfaces usually support both explicit and implicit style interactions [6, 23, 25]. Implicit style interactions allow more natural and easier to use human-computer interactions by allowing arm, hand, head, or eye movement based interactions. However, these implicit style interactions are more complex to design compared to explicit style interactions.

Table 1 summarizes the differences between characteristics of conventional and VR interfaces. As the table indicates, VR interfaces contain a richer variety and more complex types of objects, object graphics, behaviors, interactions, and communications. These characteristics bring significant challenges to the design of VR interfaces. In the absence of a conceptual model and a methodology that provide guidance during VR interface design, designers will face significant challenges in a) thinking comprehensively about the VR interface characteristics reviewed above; b) in decomposing the overall design task into smaller, conceptually distinct, and easier tasks; and c) in communicating the design to software developers. In this paper, we propose the VRID (Virtual Reality Interface Design) model and methodology to provide conceptual and methodological guidance to designers in dealing with these challenges.

VRST'01, November 15-17, 2001, Banff, Alberta, Canada.

Table 1. Comparative characteristics of conventional and VR interfaces

Characteristics	Conventional interfaces	VR interfaces
Object graphics	Mainly 2D	Mainly 3D
Object types	Mainly virtual objects	Both virtual and physical objects
Object behaviors	Mainly passive objects	Both passive and active objects
Communication patterns	Mainly simple	Mainly complex
Human-computer interactions	Mainly explicit	Both explicit and implicit

2. RELATED WORK

In this section, we briefly review relevant previous work to assess whether the existing interface design models and methodologies address the distinctive needs of VR interfaces.

A well-known user interface design model is the Four Level approach developed by Foley and colleagues [11]. This model describes the user interface as a medium that provides dialogue between the user and the computer. The four levels of the model are organized based on the meaning and the form of the dialogue between the user and the computer. The levels focus mostly on specifications of user interactions using explicit commands. This approach works well for command language and GUI (graphical user interface) style interfaces. But it is not sufficient to meet VR interface needs such as implicit interactions, object dynamism, and physical objects. Communication among objects is not sufficiently addressed either.

Another relevant user interface design model is the Command Language Grammar (CLG) developed by Moran [24]. It provides designers a model to describe and design command language interfaces. As in Foley's model, CLG divides the interface design into levels. But specifications of the levels are more formal and detailed in CLG. Although CLG works well in command language interfaces, its applicability to VR interfaces is limited. Object dynamism, interactions using implicit commands, physical objects, and communication patterns among objects are out of the scope of this model.

A third related interface design model is Shneiderman's Object-Action Interface (OAI) model [27]. It is developed particularly for design of GUI style interfaces. In order to meet the needs of GUI style interfaces, the OAI model emphasizes the importance of visual representations of objects and their actions. This model focuses on explicit command style interactions using direct manipulations, and keeps the amount of syntax small in interaction specifications. However, OAI does not address the distinctive characteristics of VR interfaces such as object dynamism, implicit style interactions, physical objects, and communication patterns among objects.

Another possibility for designers is to use general-purpose design models and methodologies such as object oriented design model and methodology and Object Modeling Technique (OMT), which are proposed for software development [5, 26]. However, these models and methodologies do not provide conceptual guidance for addressing specific challenges of VR interface design such as implicit style interactions.

Table 2 summarizes our assessment as to whether the existing interface design models and methodologies meet the distinctive needs of VR interfaces. As the table indicates, none of the four design models and methodologies that we reviewed adequately meets the distinctive needs of the VR interfaces.

Despite the lack of design models and methodologies that comprehensively address the needs of VR interfaces, there is

Table 2. Characteristics of existing design models and methodologies

Design model and methodology	Support for distinctive characteristics of VR interfaces			
	Object graphics	Object dynamism	Communication patterns	Implicit interactions
Four level	No	No	Minimal	No
CLG	No	No	No	No
OAI	Yes	No	No	No
OO	No	Yes	Yes	No

significant amount of relevant work in the VR field, which can be used as building blocks in developing such models and methodologies. The scope and complexity of VR interfaces have prompted ongoing efforts in the VR field to develop design languages, frameworks and other tools that can support the design of VR interfaces. These frameworks, languages and tools provide solutions for individual characteristics and requirements of VR interfaces such as the behavioral aspects, interaction aspects or a particular issue within these domains.

Table 3 provides a summary of the previous work addressing behavioral and interaction characteristics of VR interfaces. Once the designer figures out how to conceptualize the interface and how to decompose the overall design task into smaller components such as behaviors, interactions, communications, graphics, etc., she or he can draw on previous studies which may provide help in dealing with design challenges within individual components. However, in the absence of higher level conceptual guidance, it is difficult for designers to decompose a complex VR interface into smaller, conceptually distinct components. Therefore, there is need for design models and methodologies that provide conceptual and methodological guidance to designers at a higher level of abstraction. We propose the VRID model and methodology to address this need.

3. COMPONENTS OF A VIRTUAL REALITY SYSTEM

Before introducing the VRID model, it is important to clarify what we mean by a VR interface. As depicted in Figure 1, we conceptualize a VR system in terms of three major components: application, interface, and dialog control, inspired by the Seeheim user interface system architecture [14] and the Model View Controller architecture [20].

The application component is the VR application itself, which contains features, rules and knowledge defining the logic of the application. The interface component is the front-end through which users and other external entities exchange information with

Table 3. Previous research on behavioral and interaction characteristics of VR interfaces

	Frameworks and models	Languages	Others
Behavioral	Cremer, Kearney et al. 1995 [10], Blumberg and Galyean 1995 [4], Tu and Terzopoulos 1994 [35] Gobbetti and Balaguer 1993 [13]	Green and Halliday 1996 [15], Steed and Slater 1996 [32]	Behavioral library: Stansfield, Shawver et al. 1995 [30]
Interaction	Kessler 1999 [19], Lewis, Koved et al. 1991 [21], Gobbetti and Balaguer 1993 [13], Bowman 1999 [7]	Jacob, Deligiannidis et al. 1999 [18], Smith and Duke 1999 [28]	Interaction techniques: Bowman and Hodges 1997 [6], Liang and Green 1993 [22], Poupyrev, Billinghurst et al. 1996 [25], Stoakley, Conway et al. 1995 [33], Tanriverdi and Jacob 2000 [34], Wloka and Greenfield 1995 [36]

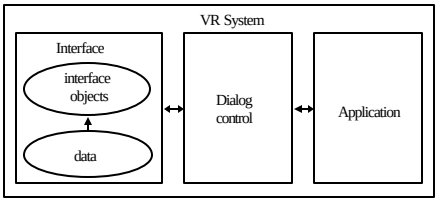


Figure 1. Components of a VR system

and manipulate the system. The interface consists of data and objects. Data refers to inputs received from users (or other external entities) whereas objects refer to entities in the interface that have well defined roles and identities. Dialog control enables communication between the application and the interface. Due to conceptual separation, internal details of application and interface components are transparent to each other. This feature allows designers to work on the two components independently.

We propose the VRID model and methodology only for the design of the interface component. Design of other VR system components is beyond the scope of this paper.

4. THE VRID MODEL

Building on our review and synthesis of the previous work on VR interface design, we identify object graphics, object behaviors, object interactions and object communications as the key constructs that designers should think about in designing VR interfaces. Therefore, we organize the VRID model around a multi-component object architecture that is depicted in Figure 2. Graphics, behavior, interaction and communicator components are included to conceptually distinguish and address the distinctive characteristics of VR interfaces. The mediator component is included to coordinate communications among the other four components of an object. These five components serve as the key constructs of our design model. Next, we will explain each of these components.

The graphics component is for specifying graphical representations of interface objects. It covers specification of all graphical models that are needed for computer-generated appearance and animations of the objects. We include the graphics component in the model in order to address the distinctive visual characteristics of VR interfaces. Since VR interface objects exhibit more complex behaviors and these behaviors need to be represented by more complex visual displays, it is important to associate object behaviors with object graphics at a high level of abstraction.

In general, responsibility for the design of visual aspects of VR interfaces lies with graphics designers rather than VR interface

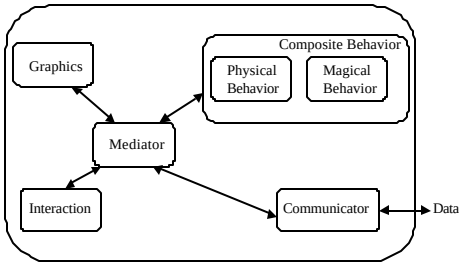


Figure 2. Multi-component object architecture used in the VRID model

designers. That is why we treat the graphics component as a black box, and focus only on its outcomes, i.e., graphical models. We aim to provide guidance to VR interface designers in specifying graphical models associated with interface objects and their behaviors. By doing so, we seek to facilitate communications between graphics designers and VR interface designers so that graphical representations and animations generated by graphics designers are compatible with behaviors specified by interface designers.

The behavior component is for specifying various types of object behaviors. In order to help designers to understand and simplify complex object behaviors, we categorize object behaviors into two groups: physical behaviors and magical behaviors. **Physical behavior** refers to those changes in an object's state that are observable in the real world. **Magical behavior** refers to those changes in an object's state, which are rarely seen, or not seen at all in the real world. Consider a virtual basketball exhibiting the following behaviors: it falls; it bounces off; it changes its color when touched by a different player; and it displays player statistics (e.g., ball possession, scores, etc.). Here, the falling and bouncing off behaviors are physical behaviors because these behaviors have counterparts in the physical world. However, "changing color" or "displaying player statistics" are magical behaviors because a real world basketball does not exhibit these behaviors.

Breaking down complex behaviors into simple physical and magical behaviors serve two purposes. First, it allows designers to generate a library of behaviors, which can be reused in creating new behaviors. Second, physical and magical distinction enables designers to assess the level of detail required in communicating design specifications to software developers unambiguously. Physical behaviors are relatively easy to describe and communicate. Since they have counterparts in the real world, software developers can easily relate to physical behaviors. Hence, interface designers may not need to describe all details of physical behaviors. Magical behaviors, on the other hand, are either rarely seen or not seen at all in the real world. Therefore,

software developers may have difficulty in visualizing the magical behaviors. Interface designers may need to specify magical behaviors in more detail to avoid any misunderstandings in the later stages of development.

Objects may exhibit **composite behaviors** that consist of a series of simple physical and magical behaviors. For example, running behavior of an athlete can be considered as a composite behavior consisting of simple physical behaviors such as leg and arm movements. By conceptually distinguishing between "simple" and "composite behaviors," we aim to help designers to decompose complex behaviors into smaller, conceptually distinct parts; and to increase the reusability of the resulting software code. Designers can combine simple behaviors in different ways and sequences to generate new composite behaviors. Breaking down a complex behavior into simpler behaviors also increases clarity of communication between designers and software developers since it is easier to visualize a series of simpler behaviors.

We devote specific attention to the design of object behaviors because defining object behaviors has typically been a challenge in VR [15]. Most virtual environments remain visually rich, but behaviorally impoverished [10]. Previous work on behavioral characteristics of VR interfaces does not provide high level guidance for decomposing complex behaviors into simpler behaviors. By proposing physical-magical and simple-composite categories, we aim to help designers to decompose complex behaviors into simpler, easier to design components, which can also be reused in creating new behaviors.

The interaction component is used to specify where inputs of the VR system come from and how they change object behaviors. The interaction component receives the input, interprets its meaning, decides on the implication of the input for object behavior, and communicates with behavioral components to make the desired change in the object behavior. VR interfaces need to support implicit style interactions, which require monitoring and interpretation of inputs such as hand, arm, head and eye movements. Each of these interaction types presents a particular design challenge. Therefore, unlike previous work, which usually merge the design of interactions and behaviors, we make a conceptual distinction between interactions and their implications for object behaviors. This distinction allows designers to focus on the challenges of interactions in the interaction component, and on the challenges of behaviors in the behavior component. It also increases reusability of resulting interactions and behaviors since interactions and behaviors are de-coupled from each other.

The mediator component is for specifying control and coordination mechanisms for communications among other components of the object. The goals are to avoid conflicts in object behaviors, and to enable loose coupling among components. To achieve these goals, we propose the mediator component by adapting the concept of "mediator design pattern" suggested by Gamma and colleagues [12]. The mediator controls and coordinates all communications within the object. When a component needs to communicate with another component, it sends its message to the mediator rather than sending it directly to the destination. This component enables designers to identify, in advance, which communication requests might lead to conflicts in object behaviors, and to specify how the requests can be managed to avoid the conflicts. Since a component only needs to know about itself and the mediator rather than having to know about all components with which it might communicate, the mediator component also ensures loose coupling between components.

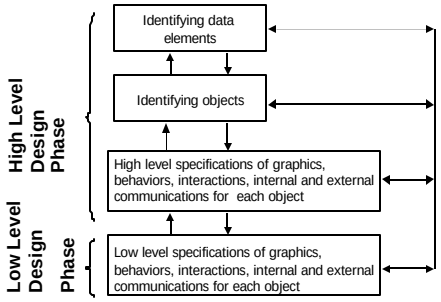


Figure 3. VRID Methodology

The communication component is for external communications of the object with other objects, data elements, or with the application component. In this component, designers need to specify sources of communication inflows into the object, destinations of communications outflows from the object, and the message passing mechanisms between them such as the synchronous, asynchronous, balking, or timeout mechanisms discussed by Booch [5]. Previous work on object behaviors discusses message-passing mechanisms among objects during low level specifications of behaviors. The difference in our approach is that we start analyzing the communication needs of objects at a higher level of abstraction, and that we make a distinction between internal and external communications of objects. By using two separate components for specification of internal and external communications mechanisms of objects, we help designers to decompose the complexity associated with design of communications into smaller, conceptually distinct components, which are easier to analyze, design, code, and maintain.

In summary, the VRID model synthesizes previous work on VR interface design and proposes a comprehensive set of modeling structures in the form of a multi-component object architecture, which helps designers to see clearly which issues and decisions are involved in VR interface design, and why.

5. THE VRID METHODOLOGY

To systematically apply the VRID model in VR interface design, we propose the VRID methodology. We conceptualize design of a VR interface as an iterative process in which requirements for the interface are translated into design specifications that can be implemented by software developers. We divide the design process into high-level and low-level design phases as depicted in Figure 3. In the high-level design phase, the goal is to specify a design solution, at a high-level of abstraction, using the multi component object architecture as a conceptual guidance. The input of the high-level design phase is a functional description of the VR interface. The output is a high-level representation of data elements and objects in the interface. Graphical models, behaviors, interactions, and internal and external communication characteristics of interface objects are identified and defined at a high level of abstraction. This output becomes the input of the low-level design phase. In the low-level design phase, the goal is to provide fine-grained details of the high-level representations, and to provide procedural details as to how they will be formally represented. The outcome of low-level design is a set of design specifications, which are represented in formal, implementation-oriented terminology, and ready to be implemented by software developers. We take a top-down approach to the design process by going from high-level abstractions to lower level details. However, this is not a linear process. It requires iterations between the high-level and low-level design phases, and reciprocal

Table 4. Description of the virtual surgery system

Consider an augmented reality system developed for training surgeons. It includes a virtual patient body and a physical biopsy needle. The surgeon wears a head-mounted display, and uses the needle to interact with the patient. A Polhemus is attached to the needle to communicate 3D coordinates of the needle to the VR system. Coordinate data is used to interpret actions of the surgeon. Abdominal area of the body is open, and organs, nerves, vessels, and muscles are visible. When the surgeon punctures an organ with the needle, it starts bleeding. The surgeon can see status of operation by prodding the organ. When prodded, the organ enlarges, and shows the status of surrounding nerves, vessels, and muscles by highlighting each of them with a unique color.

refinements at both levels of abstraction until a conceptually sound and practically implementable design emerges.

In the following sections, we explain step-by-step details of each phase of the VRID methodology. We use an example, which runs throughout the paper, to illustrate how the VRID model and methodology are applied in developing a VR interface design. The example, which is described in Table 4, is a hypothetical virtual surgery system inspired by and adapted from the descriptions given in prior studies [29, 31].

5.1 High-level (HL) design phase

High-level design phase consists of three major steps:

- HL1. Identifying data elements
- HL2. Identifying objects
- HL3. Modeling the objects
  - HL3.1. Graphics
  - HL3.2. Behaviors
  - HL3.3. Interactions
  - HL3.4. Internal communications (mediator)
  - HL3.5. External communications

5.1.1 HL1: Identifying data elements

The role of data elements is to enable communication between VR interface and entities that are external to the VR system. The goal of the first step is to identify data inflows coming into the VR interface. The interface can receive data from three sources: a) users, b) physical devices; and c) other VR systems. Designer should analyze the description of the VR interface to identify the data inflows. In the virtual surgery example, the only data element is the 3D coordinates of the needle communicated to the interface by the Polhemus. Identification of data elements is a relatively simple design task, which does not require deliberations at different levels of abstraction. We include this task in the high-level design phase in order to enable designers to understand and define data inputs of the VR interface early in the design process.

5.1.2 HL2: Identifying objects

In this step, the goal is to identify objects that have well defined roles and identities in the interface. This step involves: a) identifying potential objects mentioned in the interface description; b) deciding on legitimate objects; and c) distinguishing between virtual and physical objects. In parts (a) and (b), designers can use the object-oriented analysis and design guidelines provided for identification of potential objects and selection of legitimate objects [5, 26]. In part (c), virtual objects are those entities that need to be modeled and generated by the computer. Physical objects are physical entities that interact with the VR system. Physical objects may or may not require modeling. If they are capable of coexisting and exchanging data with the VR interface, they do not require modeling. For example, the biopsy needle in our virtual surgery example is capable of sending data to the VR interface through the Polhemus. Hence, it should be identified as a physical object. Physical objects that exhibit magical behaviors need to be identified and modeled as virtual objects. For example, a biopsy needle, which is capable of melting down and disappearing when the surgeon makes a wrong move, is exhibiting a magical behavior. This behavior is only possible through computer generation since no physical biopsy needle is capable of exhibiting this behavior. Therefore, such objects should be modeled as virtual objects.

In the virtual surgery example, potential objects are biopsy needle, patient body, organs, nerves, vessels, and muscles. In parts (a) and (b), the biopsy needle and the patient body can be identified as legitimate objects using the general guidelines of object-oriented analysis and design. The patient body is an aggregate object comprising of organ, nerve, vessel, and muscle components. In part (c), the needle can be identified as a physical object because it is capable of coexisting and exchanging data with the VR interface. The patient body should be identified as a virtual object because it exhibits magical behaviors such as highlighting nerves, vessels, and muscles with unique colors.

5.1.3 HL3: Modeling the objects

In the reminder of the design, we are no longer concerned with entities that are identified as physical objects because they do not require modeling, and their inputs to the VR system had already been identified as data elements in HL1. Therefore, the goal in this step is to model the virtual objects identified in HL2. Modeling of virtual objects involves specification of: a) graphical models; b) behaviors; c) interactions; d) internal communication characteristics; and e) external communication characteristics of the objects. Designers should analyze the interface description and use the VRID model to specify characteristics of each virtual object, as described below.

HL3.1: Graphics

In this step, the goal is to specify a high-level description of graphics needs of virtual objects. Designers should describe what kinds of graphical representations are needed for each object, and its parts, if any. Since representing objects graphically is a creative task, this step aims to provide flexibility to graphical designers by focusing only on general, high-level descriptions of graphical needs.

In our example, we should describe graphics needs of the patient body, organs, nerves, vessels, and muscles in enough detail for graphics designers to understand the context of the graphical modeling needs. We should mention that we need graphical model of an adult human body, which lies on its back on the operation table. Gender is arbitrary. Abdominal part of the body should be open, and show the organs in the area, and the nerves, vessels, and muscles that weave the organs. Boundaries of organs, nerves, vessels, and muscles must be distinguishable when highlighted.

HL3.2: Behaviors

The goals of this step are to identify behaviors exhibited by objects; classify them into simple physical, simple magical, or composite behavior categories; and to describe them in enough detail for designers to visualize the behaviors. This step involves the following activities: a) identify behaviors from the description; b) classify the behaviors into simple and composite categories; c)

classify simple behaviors into physical and magical behavior categories; and d) for composite behaviors, specify sequences in which simple behaviors are to be combined for producing the composite behaviors.

In our example, behaviors exhibited by the patient body are: 1) bleeding; 2) enlarging; 3) highlighting nerves, vessels, and muscles with unique colors; and 4) showing the status of operation. Bleeding can be specified as a simple behavior or as a composite behavior obtained by combining simple behaviors of increasing the amount, color intensity, and viscosity of blood. This design decision should be based on reusability considerations and providing clear communications with software developers. We classify bleeding as composite behavior to reuse its components in generating different behaviors of blood such as coagulating. Similarly, to prevent communication problems with software developers and help software developers to visualize what we mean by "showing the status of operation," we classify showing the status of operation as composite behavior that consists of enlarging and highlighting behaviors. Enlarging organ and highlighting nerves, vessels, and muscles are simple behaviors. Components of bleeding behavior are physical behaviors because there is nothing magic about them and they can be observed in real world. Enlarging organ and highlighting nerves, vessels, and muscles with unique colors are magical behaviors because they have no counterpart in the real world.

The next task is to specify sequences in which simple behaviors are to be combined for producing the composite behaviors. The description indicates that enlarging and highlighting behaviors should be superimposed and exhibited simultaneously. Similarly, components of the bleeding behavior are exhibited simultaneously.

HL3.3: Interactions

The goal in this step is to specify where inputs of interface objects come from and how they change object behaviors. This step involves: a) identifying interaction requests to objects; b) identifying behavioral changes caused by these requests and which behavioral components will be notified for these changes

In our example, user interacts with the patient body using the needle. The interaction component should be able to process the 3D coordinates of the needle and interpret their meaning to decide whether the surgeon is prodding or puncturing (since this is a hypothetical example, we do not specify in detail how coordinates are to be processed and interpreted). If the surgeon is prodding, the implication for the behavior of the patient body is to show the status of operation. The interaction component should communicate with the composite behavior component to initiate the "showing the status of operation" behavior. If the surgeon is puncturing, the implication for the behavior of the patient body is bleeding. The interaction component should communicate with the composite behavior component to initiate the bleeding behavior.

HL3.4: Internal communications (mediator)

In this step, the goal is to specify control and coordination needs for internal communications among the components of objects in order to avoid potential conflicts in object behavior. This involves: a) examining all communication requests and behavioral changes that are caused by these requests; b) identifying communications requests that may cause the potential conflicts; and c) deciding how to prioritize, sequence, hold or deny the communications requests to avoid the potential conflicts.

In our example, behaviors of the patient body are: 1) bleeding; 2) enlarging; 3) highlighting nerves, vessels, and muscles with unique colors; and 4) showing status of operation. If communication requests for bleeding and showing status of operation arrive simultaneously, the patient may enter into an unstable state between bleeding and showing status of operation behaviors. This conflict can be avoided by prioritizing, holding, or denying the communication requests. Here, we give higher priority to showing status of operation to avoid the conflict.

HL3.5: External communications

In this step, the goal is to specify control and coordination needs for external communications of the objects. This involves a) identifying communication inflows into the object, and their sources; b) communications outflows from the object, and their destinations; and c) describing time and buffering semantics of external communications of the object.

In our example, communication inflows into the patient body are 3D coordinates coming from the needle. There are no communications outflows. Although it is not specified in the description, for illustrative purposes, we assume that a communication between the needle and patient body starts when the needle initiates an operation (e.g., prodding, puncturing), and the patient body is ready to display the associated behavior with that operation. If the patient body is busy, the needle will wait for a specified amount of time. If the patient body is still not ready after a certain amount of time, the needle will abort the communication request.

5.2 Low-level (LL) design phase

Output of the high-level design becomes the input to the low-level design, which repeats the five modeling steps at a lower level abstraction to generate fine-grained details of the high-level design specifications:

- LL1. Graphics
- LL2. Behaviors
- LL3. Interactions
- LL4. Internal communications (mediator)
- LL5. External communications

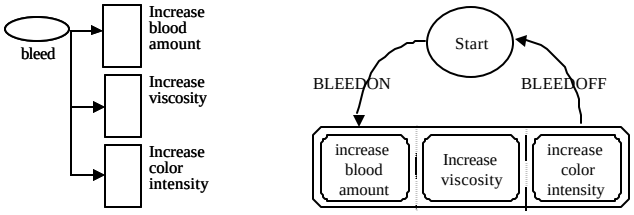
5.2.1 LL1: Graphics

Low-level design of graphics aims to associate graphical models and behaviors of objects. The outcome of this step should enable graphical designer to understand how object behaviors can be animated. This step involves matching the graphical models specified in HL3.1 with behaviors specified in HL3.2.

In our example, graphical models were specified for patient body and its parts (organs, nerves, vessels, and muscles in the abdominal area) in HL3.1. Associated behaviors specified in HL3.2 were bleeding, enlarging, highlighting, and showing the status of operation. Using the description, we need to associate graphical models of organs with all four behaviors; and the graphical models of nerves, vessels, and muscles with the bleeding and highlighting behaviors.

5.2.2 LL2: Behaviors

In low-level design of behaviors, the goal is to formalize fine-grained procedural details of behaviors that have been specified in HL3.2. Formal representation of behaviors requires use of constructs of a selected design language. In our example, we use PMIW, the user interface description language that we had originally developed for specifying interactions[18], but is also



(a) Continuous parts of bleeding behavior (b) Discrete parts of bleeding behavior

Figure 4. PMIW representation of bleeding behavior

suitable for specifying behaviors. PMIW representation requires: a) identification of discrete and continuous components of behaviors; b) use of data flow diagrams to represent continuous behaviors; and c) use of statecharts [16] and state transition diagrams to represent discrete behaviors. For illustration purposes, we depict in Figure 4 formal representation of continuous and discrete parts of the bleeding behavior using PMIW.

5.2.3 LL3: Interactions

Like low-level design of behaviors, low-level design of interactions aims to formalize fine-grained aspects of the interactions that have been specified in HL3.3. Formal representation of interactions also requires selection of a design language. PMIW is well suited for this purpose, although designers may choose any other suitable design language.

Activities outlined for formal representation of behaviors are repeated in this step, this time for representing interactions. For illustration purposes, we depict in Figure 5 a formal representation of the puncturing interaction using PMIW.

5.2.4 LL4: Internal communications (mediator)

In low-level design of internal communications, the goal is to specify scheduling mechanisms for managing the communication requests identified in HL3.4 as giving rise to conflicting object behaviors. As in the previous steps of the low level phase, designers are free to choose appropriate scheduling mechanisms. In our example, to resolve conflicts between bleeding and showing the status of operation behaviors, we prefer to use priority scheduling mechanisms that are similar to the ones used in operating systems.

5.2.5 LL5: External communications

Low-level design of external communications aims to specify the message passing mechanisms that control and coordinate external communications of the objects. Designers can select from the synchronous, asynchronous, timeout, and bulking message passing mechanisms discussed by Booch [5]. In our example, the communication needs between the patient body and the biopsy needle, which had been specified in HL3.5, can be modeled with the timeout mechanism.

This step concludes a complete pass of the phases of the VRID methodology. In applying the VRID methodology, we started with the English description of the VR system. Then, we followed the steps of the VRID methodology and applied the conceptual framework of the VRID model to decompose the overall design task into simpler, easier to design components. Each component represents a nicely encapsulated, conceptually distinct part of the interface. Designers should iterate between steps of the high and low level design phases to refine the specifications until they are convinced that conceptually sound and implementable specifications are produced.

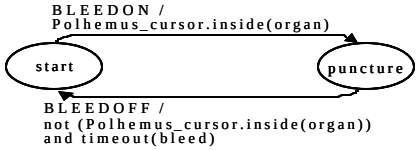


Figure 5. PMIW representation of puncturing interaction

6. DISCUSSIONS AND CONCLUSIONS

In this paper, we identified a gap in the VR literature, namely, the lack of high-level design models and methodologies for design and development of VR interfaces. By proposing the VRID model and methodology as one possible approach, we have taken an initial step towards addressing this gap.

The VRID model allows designers to think comprehensively about various types of human-computer interactions, objects, behaviors, and communications that need to be supported by VR interfaces. It enables designers to decompose the overall design task into smaller, conceptually distinct, and easier to design tasks. It provides a common framework and vocabulary, which can enhance communication and collaboration among users, designers and software developers involved in development of VR interfaces. The model may also be useful in implementation and maintenance stages of the life cycle of a VR interface since it isolates details of components, and makes changes in one component transparent to other components.

The VRID methodology contributes to practice by guiding designers in the application of the VRID model to the VR interface design process. The methodology formalizes the process of VR interface design into two phases, which represent different levels of abstraction, and breaks down the phases into a discrete number of steps. High-level design phase helps designers to conceptually design the interface without having to use implementation specific terminology. Low-level design phase guides designers in representing design specifications in formal, implementation oriented terminology. The VRID offers flexibility in selection of languages, tools or mechanisms for specifying fine-grained details of the interface.

We evaluated the VRID model and methodology by applying them in designing various types and complexities of VR interfaces, which we identified from the literature or created for test purposes, including the virtual surgery example presented here. We have also just completed an experimental user study, which assessed validity, usability, and usefulness of the VRID model and the methodology. Findings provide empirical support for the validity of the VRID model and methodology. These findings are reported in a separate paper, which is currently under review.

7. ACKNOWLEDGMENT

This work is supported by National Science Foundation Grant IRI-9625573 and Office of Naval Research Grant N00014-95-1-1099.

8. REFERENCES

[1] D. Allison, B. Wills, D. Bowman, J. Wineman, and L. F. Hodges, "The Virtual Reality Gorilla Exhibit," *IEEE Computer Graphics and Applications*, vol. 17, pp. 30-38, 1997.

[2] P. Astheimer, "A Business View of Virtual Reality," *IEEE Computer Graphics and Applications*, vol. 19, pp. 28-29, 1999.

[3] R. T. Azuma, "A Survey of Augmented Reality," *Presence*, vol. 6, pp. 355-385, 1997.

[4] B. Blumberg and T. Galyean, "Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments," presented at ACM Conference on Computer Graphics, SIGGRAPH'95, 1995.

[5] G. Booch, *Object Oriented Design with Applications*. Redwood City, CA: Benjamin/Cummings Pub. Co., 1991.

[6] D. Bowman and L. Hodges, "An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments," presented at Symposium on Interactive 3D Graphics, 1997.

[7] D. A. Bowman, "Interaction Techniques For Common Tasks In Immersive Virtual Environments: Design, Evaluation, and Application," Doctoral dissertation, Georgia Institute of Technology, 1999, http://vtopus.cs.vt.edu/~bowman/thesis/.

[8] F. P. Brooks, "What's Real About Virtual Reality?," *IEEE Computer Graphics and Applications*, vol. 19, pp. 16-27, 1999.

[9] S. Bryson, "Virtual Reality in Scientific Visualization," *Communications of the ACM*, pp. 62-71, 1996.

[10] J. Cremer, J. Kearney, and Y. Papelis, "HCSM: A Framework for Behavior and Scenario Control in Virtual Environments," *ACM Transactions on Modeling and Computer Simulation*, vol. 5, pp. 242-267, 1995.

[11] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics, Principles and Practice*, Second Edition ed. Reading, MA: Addision-Wesley, 1996.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.

[13] E. Gobbetti and J.-F. Balaguer, "VB2 An Architecture For Interaction in Synthetic Worlds," presented at ACM Symposium on User Interface Software and Technology, UIST'93, 1993.

[14] M. Green, "Report on dialogue specification tools," presented at Workshop on User Interface Management Systems, Seeheim, FRG, 1983.

[15] M. Green and S. Halliday, "A Geometric Modeling and Animation System for Virtual Reality," *Communications of the ACM*, vol. 39, pp. 46-53, 1996.

[16] D. Harel and A. Naamad, "The Statemate Semantics of Statecharts," *ACM Transactions on Software Engineering and Methodology*, vol. 5, pp. 293-333, 1996.

[17] L. Hodges, B. RothBaum, R. Kooper, O. D., T. Meyer, N. M., J. de Graff, and J. Williford, "Virtual Environments for Treating the Fear of Heights," *IEEE Computer*, vol. 28, pp. 27-34, 1995.

[18] R. J. K. Jacob, L. Deligiannidis, and S. Morrison, "A Software Model and Specification Language for Non-WIMP User Interfaces," *ACM Transactions on Computer-Human Interaction*, vol. 6, pp. 1-46, 1999.

[19] D. Kessler, "A Framework for Interactors in Immersive Virtual Environments," presented at IEEE Virtual Reality, 1999.

[20] G. E. Krasner and S. T. Pope, "A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80," *Journal of Object-Oriented Programming*, vol. 1, pp. 26-49, 1988.

[21] J. B. Lewis, L. Koved, and D. T. Ling, "Dialogue Structures for Virtual Worlds," presented at ACM Human Factors in Computing Systems, CHI'91, 1991.

[22] J. Liang and M. Green, "Interaction Techniques For A Highly Interactive 3D Geometric Modeling System," presented at ACM Solid Modeling, 1993.

[23] M. Mine, F. Brooks, and C. Sequin, "Moving Objects in Space: Exploiting Proprioception in Virtual Environment Interaction," presented at SIGGRAPH'97, Los Angeles, CA, 1997.

[24] T. Moran, "The Command Language Grammar," *International Journal of Man-Machine Studies*, vol. 15, pp. 3-50, 1981.

[25] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa, "The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR," presented at ACM Symposium on User Interface Software and Technology, UIST'96, 1996.

[26] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson, *Object Oriented Modeling and Design*. Edglewood Cliffs, New Jersey: Prentice Hall, 1991.

[27] B. Shneiderman, *Designing the User Interface, Strategies for Effective Human-Computer Interaction*, Third Edition ed. Reading, MA: Addison-Wesley, 1998.

[28] S. Smith and D. J. Duke, "Virtual Environments as Hybrid Systems," presented at Annual Conference Eurographics UK Chapter, Cambridge, 1999.

[29] D. Sorid and S. Moore, "The Virtual Surgeon," *IEEE Spectrum*, pp. 26-31, 2000.

[30] S. Stansfield, D. Shawver, N. Miner, and D. Rogers, "An Application of Shared Virtual Reality to Situational Training," presented at IEEE Virtual Reality Annual International Symposium, 1995.

[31] A. State, M. A. Livingston, G. Hirota, W. F. Garrett, M. C. Whitton, H. Fuchs, and E. D. Pisano, "Technologies for Augmented-Reality Systems: realizing Ultrasound-Guided Needle Biopsies," presented at ACM Conference on Computer Graphics, SIGGRAPH'96, 1996.

[32] A. Steed and M. Slater, "A Dataflow Representation for Defining Behaviors within Virtual Environments," presented at IEEE Virtual Reality Annual Symposium, VRAIS'96, 1996.

[33] R. Stoakley, M. Conway, and R. Pausch, "Virtual Reality on a WIM: Interactive Worlds in Miniature," presented at ACM Human Factors in Computing Systems, CHI'95, 1995.

[34] V. Tanriverdi and R. J. K. Jacob, "Interacting with Eye Movements in Virtual Environments," presented at ACM Human Factors in Computing Systems, CHI'00, 2000.

[35] X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior," presented at ACM Conference on Computer Graphics, SIGGRAPH'94, 1994.

[36] M. Wloka and E. Greenfield, "The Virtual Tricorder: A Uniform Interface for Virtual Reality," presented at ACM Symposium on User Interface Software and Technology, UIST'95, 1995.