Usability Evaluation in Virtual Environments: Classification and Comparison of Methods

Abstract

Virtual environments (VEs) are a relatively new type of human-computer interface in which users perceive and act in a three-dimensional world. The designers of such systems cannot rely solely on design guidelines for traditional two-dimensional interfaces, so usability evaluation is crucial for VEs. We present an overview of VE usability evaluation. First, we discuss some of the issues that differentiate VE usability evaluation from evaluation of traditional user interfaces such as GUIs. We also present a review of VE evaluation methods currently in use, and discuss a simple classification space for VE usability evaluation methods. This classification space provides a structured means for comparing evaluation methods according to three key characteristics: involvement of representative users, context of evaluation, and types of results produced. To illustrate these concepts, we compare two existing evaluation approaches: testbed evaluation [Bowman, Johnson, & Hodges, 1999], and sequential evaluation [Gabbard, Hix, & Swan, 1999]. We conclude by presenting novel ways to effectively link these two approaches to VE usability evaluation.

1 Introduction and motivation

During the past several years, virtual environments (VEs) have gained broad attention throughout the computing community. During roughly that same time period, usability has become a major focus of interactive system development. *Usability* can be broadly defined as "ease of use" plus "usefulness", including such quantifiable characteristics as learnability, speed

and accuracy of user task performance, user error rate, and subjective user satisfaction [Hix & Hartson 1993; Shneiderman 1992]. Despite intense and widespread research in both VEs and usability, until recently there were very few examples of research coupling VE technology with usability — a necessary coupling if VEs are to reach their full potential. By focusing on usability from the very beginning of the development process, developers are more likely to avoid creating interaction techniques (ITs) that do not match appropriate user task requirements or to avoid producing standards and principles for VE user interface development that are nonsensical. In this paper, we focus on *usability evaluation* of VEs – determining how different ITs, interface styles, and numerous other factors such as information organization, visualization, and navigation affect the usability of VE applications and user interface components.

Although numerous methods exist for usability evaluation of interactive computer applications, these methods have well-known limitations, especially for evaluating VEs. For example, most usability evaluation methods are applicable only to a narrow range of interface types (e.g., graphical user interfaces, or GUIs) and have had little or no use with innovative, non-routine interfaces such as those found in VEs. VE applications have interaction styles so radically different from ordinary user interfaces that well-proven methods that produce usable GUIs may be neither appropriate nor effective.

There have been attempts to adapt traditional usability evaluation methods for use in VEs, and a few notable efforts to develop structured usability evaluation methods for VEs. In this paper, we present a survey of existing approaches to usability evaluation to VEs. We begin by making explicit some of the important differences between evaluation of VE user interfaces and traditional GUIs. Next, we categorize usability evaluation methods based on three important characteristics: involvement of representative users, context of evaluation, and types of results

produced. Two major approaches are presented and compared: *testbed evaluation*, which focuses on low-level ITs in a generic context, and *sequential evaluation,* which applies several different types of evaluation methods within the context of a particular VE application. Finally, we present links between these two approaches that provide more power and an even broader set of methods to VE developers and researchers.

We would like to set the context for this paper by explaining some terminology. First, we take a broad approach to assessing *usability*: it includes any characteristic relating to the ease of use and usefulness of an interactive software application, including user task performance, subjective satisfaction, user comfort, and so on. We define *usability evaluation* as assessment of a specific application's user interface (often at the prototype stage), an interaction metaphor or technique, or an input device, for the purpose of determining its actual or probable usability. *Usability engineering* is, in general, a term covering the entire spectrum of user interaction development activities, including user and task analysis, conceptual and detailed user interaction design, prototyping, and numerous methods of usability evaluation. The roles involved in usability evaluation typically include a *developer* (who implements the application and/or interface software), an *evaluator* (who conducts evaluation sessions), and a *user* or *subject* (who participates in evaluation sessions). Finally, we consider "VEs" to include a broad range of systems, from interactive stereo graphics on a monitor to a fully-immersive 6-sided CAVE™. Most of the distinctive aspects of VE evaluation (section 2), however, stem from the use of partially- or fully-immersive systems.

2 Distinctive characteristics of VE evaluation

The approaches we present for usability evaluation of virtual environments have been developed and used in response to perceived differences between the evaluation of VEs and the

evaluation of traditional user interfaces such as GUIs. Many of the fundamental concepts and goals are similar, but use of these approaches in the context of VEs is distinct. Here, we present some of the issues that differentiate VE usability evaluation, organized into several categories. The categories contain overlapping considerations, but provide a rough partitioning of these important issues. Note that many of these issues cannot be found in the literature, but instead come from personal experience and extensive discussions with colleagues.

2.1 Physical environment issues

One of the most obvious differences between VEs and traditional interfaces is the *physical environment* in which the interface is used. In VEs, non-traditional input and output devices are used, which can preclude the use of some types of evaluation. Users may be standing rather than sitting, and they may be moving about a large space, using whole-body movements. These properties give rise to several issues for usability evaluation. Following are some examples:

- In interfaces using non-see-through head-mounted displays (HMDs), the user cannot see the surrounding physical world. Therefore, the evaluator must ensure that the user will not bump into walls or other physical objects, trip over cables, or move outside the range of the tracking device. A related problem in surround-screen VEs (such as the CAVE™) is that the physical walls can be difficult to see because of projected graphics. Problems of this sort could contaminate the results of a usability evaluation (e.g., if the user trips while in the midst of a timed task), and more importantly could cause injury to the user. To mitigate risk, the evaluator can ensure that cables are bundled and will not get in the way of the user (e.g., cables may descend from above). Also, the user may be placed in a physical enclosure that limits movement to areas where there are no physical objects to interfere.

- Many VE displays do not allow multiple simultaneous viewers (e.g., user and evaluator), so equipment must be set up so that an evaluator can see the same image as the user. With an HMD, for example, this can be done by splitting the video signal and sending it to both the HMD and a monitor. In a surround-screen or workbench VE, a monoscopic view of the scene could be rendered to a monitor, or, if performance will not be adversely affected, both the user and the evaluator can be tracked (this can cause other problems, however – see section 2.2 on evaluator considerations). If images are viewed on a monitor, then it is difficult to see both the actions of the user and the graphical environment at the same time, meaning that multiple evaluators may be necessary to observe and collect data during an evaluation session.

- A common and very effective technique for generating important qualitative data during usability evaluation sessions is the "think aloud" protocol. With this technique, subjects talk about their actions, goals, and thoughts regarding the interface while they are performing specific tasks. In some VEs, however, voice recognition is used as an IT, rendering the think aloud protocol much more difficult and perhaps even impossible. Post-session interviews may help to recover some of the information that would have been obtained from the think aloud protocol.

- Another common technique involves recording video of both the user and the interface. Since VE users are often mobile, a single, fixed camera may require a very wide shot, which may not allow precise identification of actions. This could be addressed by using a tracking camera (additional expense and complexity) or a camera operator (additional personnel). Moreover, views of the user and the graphical environment must be

synchronized so that cause and effect can clearly be seen on the videotape. Finally, the problem of recording video of a stereoscopic graphics image must be overcome.

- An ever-larger number of proposed VE applications are shared among two or more users. These collaborative VEs become even more difficult to evaluate than single-user VEs because of physical separation between users (i.e., different users are in more than one physical location), the additional information that must be recorded for each user, the unpredictability of network behavior as a factor influencing usability, the possibility that each user will have different input and output devices, and the additional complexity of the system, which may cause more frequent crashes or other problems.

2.2 Evaluator issues

A second set of issues relates to the role of the evaluator in a VE usability evaluation. Because of the complexities and distinctive characteristics of VEs, a usability study may require multiple evaluators, different evaluator roles and behaviors, or both. Following are some examples:

- Many VEs attempt to produce a sense of *presence* in the user; that is, a feeling of actually being in the virtual world rather than the physical one. Evaluators can cause breaks in presence if the user can sense them. In VEs using projected graphics, the user will see an evaluator if the evaluator moves into the user's field of view. This is especially likely in a CAVE™ environment where it is difficult to see the front of a user (e.g., their facial expressions and detailed use of handheld devices) without affecting their sense of presence. This may break presence since the evaluator is not part of the virtual world. In any type of VE, touching or talking to the user can cause such breaks. If the evaluation is measuring presence, or if presence is hypothesized to affect

performance on the task being evaluated, then the evaluator must take care to remain unsensed during the evaluation.

- Because breaks in presence are so important, an evaluator probably does not wish to intervene at all during an evaluation session. This means that the experimental application/interface must be robust and bug-free, so that the session does not have to be interrupted to fix a problem. Also, instructions given to the user must be very detailed, explicit, and precise, and the evaluator should make sure the user has a complete understanding of the procedure and tasks before beginning the session.
- VE hardware and software are often more complex and less robust than traditional user interface hardware and software. Again, multiple evaluators may be needed to do tasks such as helping the user with display and input hardware, running the software that produces graphics and other output, recording data such as timings and errors, and recording critical incidents and other qualitative observations of a user's actions.
- Traditional user interfaces typically require only a discrete, single stream of input (e.g., from mouse and keyboard), but many VEs include multi-modal input, combining discrete events, gestures, voice, and/or whole-body motion. It is much more difficult for an evaluator to process these multiple input streams simultaneously and record an accurate log of the user's actions. These challenges make multiple evaluators and video even more important.

2.3 User issues

There are also a large number of issues related to the user population used as subjects in VE usability evaluations. In traditional evaluations, subjects are gleaned from the target user

population of an application or from a similar representative group of people. Efforts are often made, for example, to preserve gender equity, to have a good distribution of ages, and to test both experts and novices, if these differences are representative of the target user population. The nature of VE evaluation, however, does not always allow for such straightforward selection of users. Following are some examples:

- VEs are still often a "solution looking for a problem." Because of this, the target user population for a VE application or IT to be evaluated may not be known or well-understood. For example, a study comparing two virtual travel techniques is not aimed at a particular set of users. Thus, it may be difficult to generalize performance results. The best course of action is to evaluate the most diverse user population possible in terms of age, gender, technical ability, physical characteristics, and so on, and to include these factors in any models of performance.

- It may be impossible to differentiate between novice and expert users, since there are very few potential subjects who could be considered experts in VEs. Most users who could be considered experts might be, for example, research staff, whose participation in an evaluation could confound the results. Also, because most users are typically novices, the evaluation itself may need to be framed at a lower cognitive and physical level. Evaluators can make no assumptions about a novice user's ability to understand or use a given IT or device.

- Because VEs will be novel to many potential subjects, the results of an evaluation usually exhibit high variability and differences among individuals. This means that the number of subjects needed to obtain a good picture of performance may be higher than for traditional usability evaluations. If statistically significant results are required

(depending on the type of usability evaluation being performed), the number of subjects may be even greater.

- Researchers are still studying a large design space for VE ITs and devices. Because of this, evaluations often compare two or more techniques, devices, or combinations of the two. To perform such evaluations using a within-subjects design, users must be able to adapt to a wide variety of situations. If a between-subjects design is used, a larger number of subjects will again be needed.

- VE evaluations must consider the effects of simulator sickness and fatigue on subjects. Although some of the causes of simulator sickness are known, there are still no predictive models for simulator sickness [Kennedy, Stanney, and Dunlap, 2000], and little is known regarding acceptable exposure time to VEs. For evaluations, then, a worst-case assumption must be made. A lengthy experiment (anything over 30 minutes might be considered lengthy) must contain planned rest breaks and contingency plans in case of ill or fatigued subjects. Shortening the experiment is often not an option, especially if statistically significant results are needed.

- Because we do not know exactly what VE situations cause sickness or fatigue, most VE evaluations should include some measurement (e.g., subjective, questionnaire-based [e.g., Kennedy et al., 1993], or physiological) of these factors. A result indicating that an IT was 50 percent faster than any other evaluated technique would be severely misleading if that IT also made 30 percent of subjects sick! Thus, user comfort measurements should be included in low-level VE evaluations.

- Presence is another example of a measure often required in VE evaluations that has no analogue in traditional user interface evaluation. VE evaluations must often take into account subjective reports of perceived presence, perceived fidelity of the virtual world, and so on. Questionnaires [e.g., Witmer & Singer, 1998] have been developed that purportedly obtain reliable and consistent measurements of such factors.

2.4 Issues related to type of usability evaluation

Traditional usability evaluation can take many forms. These include informal user studies, formal experiments, task-based usability studies, heuristic evaluations, and the use of predictive models of performance (see section 3 for further discussion of these types of evaluations). There are several issues related to the use of various types of usability evaluation in VEs. Following are some examples:

- Evaluations based solely on heuristics (i.e., design guidelines), performed by usability experts, are very difficult in VEs because of a lack of published, verified guidelines for VE user interface design. There are some notable exceptions [Bowman, 2001; Gabbard, 1997; Kaur, 1998], but for the most part it is difficult to predict the usability of a VE interface without studying real users attempting representative tasks in the VE. It is not likely that a large number of heuristics will appear at least until VE input and output devices become standardized. Even assuming standardized devices, however, the design space for VE ITs and interfaces is very large, making it difficult to produce effective and general heuristics to use as the basis for evaluation.

- Another major type of usability evaluation that does not employ users is the application of performance models (e.g., GOMS, Fitts' Law). Again, such models simply do not

exist at this stage of VE development. However, the lower cost of both heuristic evaluation and performance model application makes them attractive for evaluation.

- Because of the complexity and novelty of VEs, the applicability or utility of automated, tool-based evaluation may be greater than it is for more traditional user interfaces. For example, several issues above have noted the need for more than one evaluator in a VE usability evaluation session. Automated usability evaluations could reduce the need for several evaluators in a single session. There are at least two possibilities for automated usability evaluation of VE user interfaces: first, to automatically collect and/or analyze data generated by one or more users in a VE, and second, to perform an analysis of an interface design using an interactive tool that embodies design guidelines (similar to heuristics). Some work has been done on automatic collection and analysis of data using specific types of repeating patterns in users' data as indicators of potential usability problems (e.g., [Siochi & Hix, 1991]). However this work was performed on a typical GUI, and there appears to be no research yet conducted that studies automated data collection and evaluation of users' data in VEs. Thus, differences in use of these kinds of data for VE usability evaluation have not been explored, but they would involve, at a minimum, collating data from multiple users in a single session, possibly at different physical locations and even in different parts of the VE. At least one tool, MAUVE (Multi-Attribute Usability evaluation tool for Virtual Environments) incorporates design guidelines organized around several VE categories such as navigation, object manipulation, input, output (e.g., visual, auditory, haptic), and so on [Stanney, personal communication]. Within each of these categories, MAUVE presents a series of questions to an evaluator, who uses the tool to perform a multi-criteria heuristic-style evaluation of

a specific VE user interface. Further work in both of these types of automated usability evaluation is of interest, especially in light of the expense of developing and evaluating VEs.

- When performing formal experiments to quantify and compare the usability of various VE ITs, input devices, interface elements, and so on, it is often difficult to know which factors have a potential impact on the results. Besides the primary independent variable (e.g., a specific IT), there are a large number of other potential factors that could be included, such as environment, task, system, or user characteristics. One approach is to try to vary as many of these potentially important factors as possible during a single experiment. Such "testbed evaluation" [Bowman, Johnson, & Hodges, 1999] (see Section 3.2) has been done with some success. The other extreme would be to simply hold as many of these other factors as possible constant, and evaluate only in a particular set of circumstances. Thus, formal VE experimental evaluations may be either overly simplistic or overly complex – finding the proper balance is difficult.

2.5 Other issues

- VE usability evaluations generally focus at a lower level than traditional user interface evaluations. In the context of GUIs, a standard look and feel and a standard set of interface elements and ITs exist, so evaluation usually looks at subtle interface nuances or overall interface metaphors. In the VE field, however, there are no interface standards, and we do not have a good understanding of the usability of various interface types. Therefore, VE evaluations most often compare lower-level components, such as ITs or input devices.

- It is tempting to over-generalize the results of evaluations of VE interaction performed in a generic (non-application) context. However, because of the fast-changing and complex nature of VEs, one cannot assume anything (display type, input devices, graphics processing power, tracker accuracy, etc.) about the characteristics of a real VE application. Everything has the potential to change. Therefore, it is important to include information about the environment in which the evaluation was performed, and to evaluate in a range of environments (e.g., using different devices) if possible.

3 Current evaluation methods

A review of recent VE literature indicates that a growing number of researchers and developers are considering usability at some level. Some are employing extensive usability evaluation techniques with a carefully-chosen, representative user base (e.g., [Hix et al., 1999]), while others undertake efforts that do not involve users, such as review and inspection by a usability expert (e.g., [Steed & Tromp, 1998]).

From the literature, we have compiled a list of usability evaluation methods that have been applied to VEs. Most of these methods were developed for 2D or GUI usability evaluation and have been subsequently extended to support VE evaluation. These methods include:

- *Cognitive Walkthrough*: an approach to evaluating a user interface based on stepping through common tasks that a user would perform and evaluating the interface's ability to support each step. This approach is intended especially to help understand the usability of a system for first-time or infrequent users, that is, for users in an exploratory learning mode [Polson et al., 1992].
- *Formative Evaluation* (both formal and informal): an observational, empirical evaluation method that assesses user interaction by iteratively placing representative users in task-based

scenarios in order to identify usability problems, as well as to assess the design's ability to support user exploration, learning, and task performance [Scriven, 1967; Hix & Hartson, 1993]. Formative evaluations can range from being rather informal, providing mostly qualitative results such as critical incidents, user comments, and general reactions, to being very formal and extensive, producing both qualitative and quantitative (e.g., task timing, errors, etc.) results.

- *Heuristic or Guidelines-Based Expert Evaluation*: a method in which several usability experts separately evaluate a user interface design (probably a prototype) by applying a set of "heuristics" or design guidelines that are relevant. No representative users are involved. Results from the several experts are then combined and ranked to prioritize iterative (re)design of each usability issue discovered [Nielsen & Mack, 1994].
- *Post-hoc Questionnaire*: a written set of questions used to obtain demographic information and views and interests of users after they have participated in a (typically formative) usability evaluation session. Questionnaires are good for collecting subjective data and are often more convenient and more consistent than personal interviews [Hix & Hartson, 1993].
- *Interview / Demo*: a technique for gathering information about users by talking directly to them. An interview can gather more information than a questionnaire and may go into a deeper level of detail. Interviews are good for getting subjective reactions, opinions, and insights into how people reason about issues. "Structured interviews" have a pre-defined set of questions and responses. "Open-ended interviews" permit the respondent (interviewee) to provide additional information, ask broad questions without a fixed set of answers, and explore paths of questioning which may occur to the interviewer spontaneously during the

interview [Hix & Hartson, 1993]. Demonstrations (typically of a prototype) may be used in conjunction with user interviews to aid a user in talking about the interface.

- *Summative* or *Comparative Evaluation* (both formal and informal): an evaluation and statistical comparison of two or more configurations of user interface designs, user interface components, and/or user ITs [Scriven, 1967; Hix & Hartson, 1993]. As with formative evaluation, representative users perform task scenarios as evaluators collect both qualitative and quantitative data. As with formative evaluations, summative evaluations can be formally or informally applied.

There have been several innovative approaches to evaluating VEs that employ one or more of the evaluation methods given above. Some of these approaches are shown in Table 1. This particular set of research literature was chosen to illustrate the wide range of methods and combination of methods available for use.

Research Example	***Usability Evaluation Method(s) Employed***
[Bowman & Hodges, 1997]	Informal Summative
[Bowman, Johnson, & Hodges, 1999]	Formal Summative, Interview
[Darken & Sibert, 1996]	Summative Evaluation, Post-hoc Questionnaire
[Gabbard, Hix & Swan, 1999] [Hix et. al, 1999]	User Task Analysis, Heuristic Evaluation, Formative Evaluation, Summative Evaluation
[Steed & Tromp, 1998]	Heuristic Evaluation, Cognitive Walkthrough
[Slater, Usoh & Steed, 1995]	Post-hoc Questionnaire

Table 1. Examples of VE usability evaluation from the literature

A closer look at these, and other research efforts, shows that the type of evaluation method(s) used, as well as the manner in which it was extended or applied, varies from study to study. It is

not clear whether an evaluation method or set of methods can be reliably and systematically prescribed given the wide range of design goals and user interfaces inherent in VEs. However, it is possible to classify those methods that have been applied to VE evaluation to reveal common and distinctive characteristics among methods.

3.1 Classification of VE usability evaluation methods

We have created a novel classification space for VE usability evaluation methods. The classification space (figure 1) provides a structured means for comparing evaluation methods according to three key characteristics: *involvement of representative users*, *context of evaluation*, and *types of results produced*.

The first characteristic discriminates between those methods that *require* the participation of representative users (to provide design or use-based experiences and options), and those methods that do not (methods not requiring users still require a usability expert). The second characteristic describes the type of context in which the evaluation takes place. In particular, this characteristic identifies those methods that are applied in a generic context and those that are applied in an application-specific context. The context of evaluation inherently imposes restrictions on the applicability and generality of results. Thus, conclusions or results of evaluations conducted in a generic context can typically be applied more broadly (i.e., to more types of interfaces) than results of an application-specific evaluation method, which may be best-suited for applications that are similar in nature. The third characteristic identifies whether or not a given usability evaluation method produces (primarily) qualitative or quantitative results.

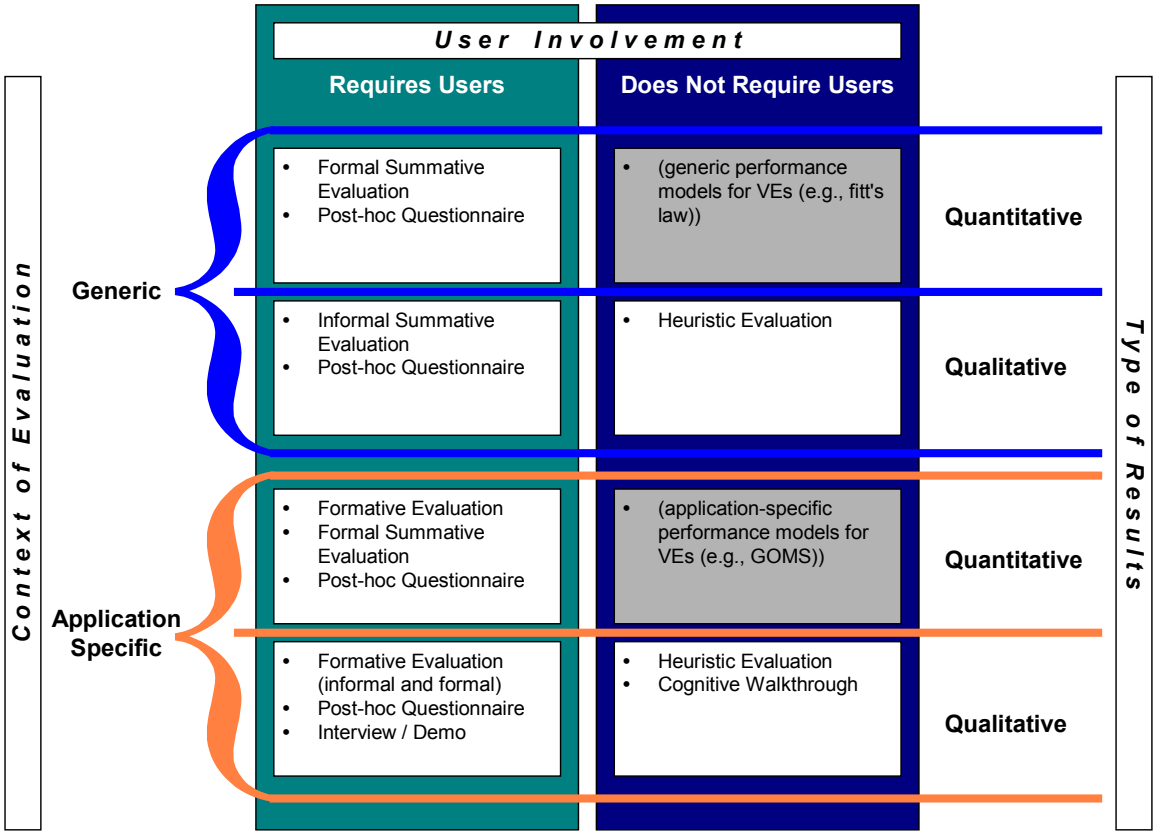

Figure 1. A Classification of Usability Evaluation Methods for VEs

Note that the characteristics described above are not designed to be mutually exclusive, and are instead designed to convey one (of many) usability evaluation method characteristics. For example, a particular usability evaluation method may produce both quantitative and qualitative results. Indeed, many of the identified methods are flexible enough to provide insight at many levels. We chose these three characteristics (over other potential characteristics) because they are often the most significant (to evaluators) due to their overall effect on the usability process. That is, a researcher interested in undertaking usability evaluation will likely need to know what the evaluation will cost, what the impact of the evaluation will be, and how the results can be applied. Each of the three characteristics address these concerns; the degree of user involvement directly affects the cost to proctor and analyze the evaluation, the results of the process indicate

what type of information will be produced (for the given cost), and the context of evaluation inherently dictates to what extent results may be applied.

This classification is useful on several levels. It structures the space of evaluation methods, and provides a practical vocabulary for discussion of methods in the research community. It also allows one to compare two or more methods and understand how they are similar or different on a fundamental level. Finally, it reveals holes in the space [Card, Mackinlay, and Robertson, 1990] – combinations of the three characteristics that have not yet been tried in the VE community.

Figure 1 shows that there are two such holes in our space (the shaded boxes). More specifically, there appear to be no current VE usability evaluation methods that do not require users and that can be applied in a generic context to produce quantitative results (upper right of figure 1). Note that some possible existing 2D and GUI evaluation methods are listed in parentheses, but these have not yet (to our knowledge) been applied to VEs. Similarly, there appears to be no method that provides quantitative results in an application-specific setting that does not require users (third box down on the right of figure 1). These areas may be interesting avenues for further research.

A shortcoming of the classification is that it does not convey "when" in the software development lifecycle a method is best applied, or "how" several methods may be applied either in parallel or serially. In most cases, the answers to these questions cannot be answered without a comprehensive understanding of each of the methods presented, as well as the specific goals and circumstances of the research or development effort. In the following sections, we present two well-developed VE evaluation approaches, compare them in terms of practical usage and results, and discuss ways they can be linked for greater power and efficiency.

3.2 Testbed evaluation approach

Bowman and Hodges [1999] take the approach of empirically evaluating ITs outside the context of applications (i.e., within a generic context, rather than within a specific application), and add the support of a framework for design and evaluation, which we summarize here. Principled, systematic design and evaluation frameworks give formalism and structure to research on interaction, rather than relying solely on experience and intuition. Formal frameworks provide us not only with a greater understanding of the advantages and disadvantages of current techniques, but also with better opportunities to create robust and well-performing new techniques, based on knowledge gained through evaluation. Therefore, this approach follows several important evaluation concepts, elucidated in the following sections. Figure 2 presents an overview of this approach.

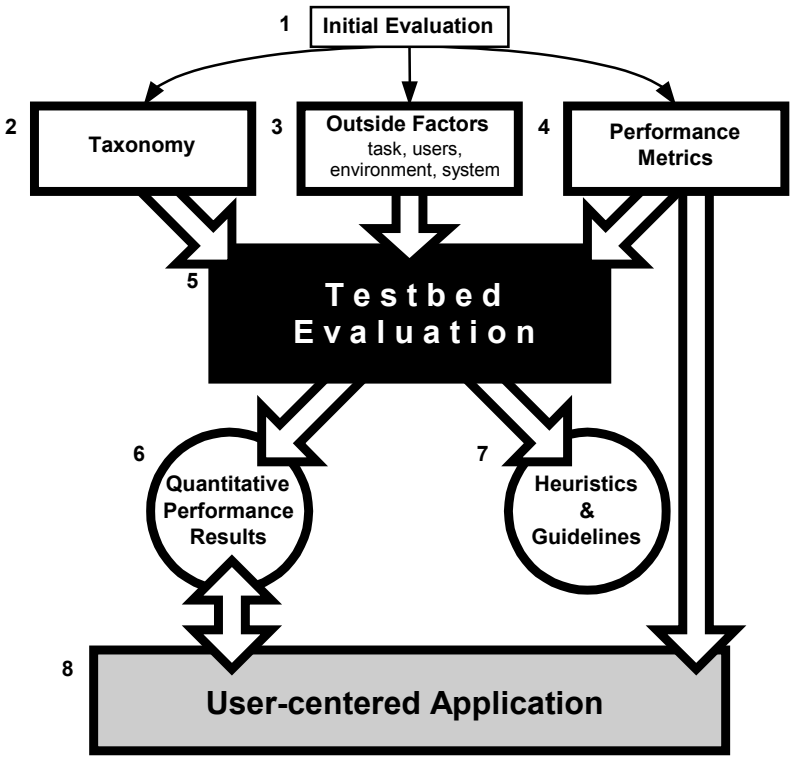

Figure 2. Bowman & Hodges' [1999] Evaluation Approach

The first step towards formalizing the design, evaluation, and application of ITs is to gain an intuitive understanding of the generic interaction tasks in which one is interested, and current techniques available for the tasks (see figure 2, area labeled 1). This is accomplished through experience using ITs and through observation and evaluation of groups of users. These initial evaluation experiences are heavily drawn upon for the processes of building a taxonomy, listing outside influences on performance, and listing performance measures. It is helpful, therefore, to gain as much experience of this type as possible so that good decisions can be made in the next phases of formalization.

The next step is to establish a *taxonomy* (figure 2, 2) of ITs for the interaction task being evaluated. These taxonomies partition a task into separable subtasks, each of which represents a decision that must be made by the designer of a technique. In this sense, a taxonomy is the product of a careful task analysis. Once the task has been decomposed to a sufficiently fine-grained level, the taxonomy is completed by listing possible technique components for accomplishing each of the lowest-level subtasks. An IT is made up of one technique component from each of the lowest-level subtasks. For example, the task of changing an object's color might be made up of three subtasks: selecting an object, choosing a color, and applying the color. The subtask for choosing a color might have two possible technique components: changing the values of R, G, and B sliders, or touching a point within a 3D color space. The subtasks and their related technique components make up a taxonomy for the object coloring task.

Ideally, the taxonomies established by this approach need to be correct, complete, and general. Any IT that can be conceived for the task should fit within the taxonomy. Thus, subtasks will necessarily be abstract. The taxonomy will also list several possible technique components for each of the subtasks, but they do not list every conceivable component.

Building taxonomies is a good way to understand the low-level makeup of ITs, and to formalize differences between them, but once they are in place, they can also be used in the design process. One can think of a taxonomy not only as a characterization, but also as a design space. Since a taxonomy breaks the task down into separable subtasks, a wide range of designs can be considered quickly, simply by trying different combinations of technique components for each of the subtasks. There is no guarantee that a given combination will make sense as a complete IT, but the systematic nature of the taxonomy makes it easy to generate designs and to reject inappropriate combinations.

ITs cannot be evaluated in a vacuum. A user's performance on an interaction task may depend on a variety of factors (figure 2, 3), of which the IT is but one. In order for the evaluation framework to be complete, such factors must be included explicitly, and used as secondary independent variables in evaluations. Bowman and Hodges identified four categories of outside factors.

First, *task characteristics* are those attributes of the task that may affect user performance, including distance to be traveled or size of the object being manipulated. Second, the approach considers *environment characteristics*, such as the number of obstacles and the level of activity or motion in the VE. *User characteristics*, including cognitive measures such as spatial ability or physical attributes such as arm length, may also contribute to user performance. Finally, *system characteristics* may be significant, such as the lighting model used or the mean frame rate.

This approach is designed to obtain information about human performance in common VE interaction tasks – but what is performance? Speed and accuracy are easy to measure, are quantitative, and are clearly important in the evaluation of ITs, but there are also many other performance metrics (figure 2, 4) to be considered. Thus, this approach also considers more

subjective performance values, such as perceived ease of use, ease of learning, and user comfort. For VEs in particular, presence [Witmer & Singer, 1998] might be a valuable measure. The choice of IT could conceivably affect all of these, and they should not be discounted. Also, more than any other current computing paradigm, VEs involve the user's senses and body in the task. Thus, a focus on user-centric performance measures is essential. If an IT does not make good use of human skills, or if it causes fatigue or discomfort, it will not provide overall usability despite its performance in other areas.

Bowman and Hodges [1999] use *testbed evaluation* (figure 2, 5) as the final stage in the evaluation of ITs for VE interaction tasks. This approach allows generic, generalizable, and reusable evaluation through the creation of testbeds – environments and tasks that involve all important aspects of a task, that evaluate each component of a technique, that consider outside influences (factors other than the IT) on performance, and that have multiple performance measures. A testbed experiment uses a formal, factorial, experimental design, and normally requires a large number of subjects. If many ITs or outside factors are included in the evaluation, the number of trials per subject can become overly large, so ITs are usually a between-subjects variable (each subject uses only a single IT), while other factors are within-subjects variables. Testbed evaluations have been performed for the tasks of travel and selection/manipulation [Bowman, Johnson, and Hodges, 1999].

Testbed evaluation produces a set of results or models (figure 2, 6) that characterize the usability of an IT for the specified task. Usability is given in terms of multiple performance metrics, with respect to various levels of outside factors. These results become part of a performance database for the interaction task, with more information being added to the database

each time a new technique is run through the testbed. These results can also be generalized into heuristics or guidelines (figure 2, 7) that can easily be evaluated and applied by VE developers.

The last step is to apply the performance results to VE applications (figure 2, 8), with the goal of making them more useful and usable. In order to choose ITs for applications appropriately, one must understand the interaction requirements of the application. There is no single "best" technique, because the technique that is best for one application will not be optimal for another application with different requirements. Therefore, applications need to specify their interaction requirements before the most appropriate ITs can be chosen. This specification is done in terms of the performance metrics that have already been defined as part of the formal framework. Once the requirements are in place, the performance results from testbed evaluation can be used to recommend ITs that meet those requirements.

3.3 Sequential evaluation approach

Gabbard, Hix & Swan [1999] present a sequential approach to usability evaluation for specific VE applications. The sequential evaluation approach is a usability engineering approach, and addresses both design and evaluation of VE user interfaces. However, for the scope of this paper, we focus on different types of evaluation and address analysis, design, and prototyping only when they have a

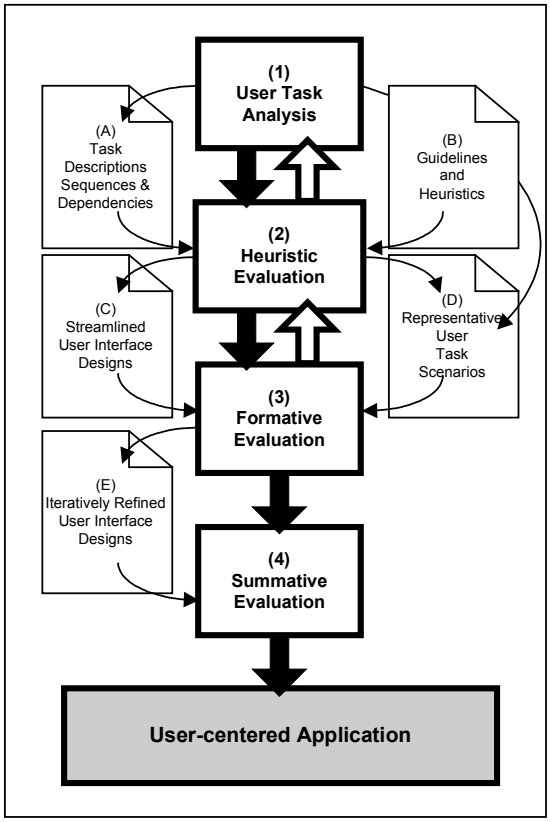

Figure 3. Gabbard, Hix & Swan's [1999] Sequential Evaluation Approach

direct effect on evaluation.

While some of its components are well-suited for evaluation of generic ITs, the complete sequential evaluation approach employs application-specific guidelines, domain-specific representative users, and application-specific user tasks to produce a usable and useful interface for a particular application. In many cases, results or lessons learned may be applied to other, similar applications (for example, VE applications with similar display or input devices, or with similar types of tasks) and, in other cases (albeit less often), it is possible to abstract the results to generic cases.

Sequential evaluation evolved from iteratively adapting and enhancing existing 2D and GUI usability evaluation methods. In particular, we modified and extended specific methods to account for complex ITs, non-standard and dynamic user interface components, and multimodal tasks inherent in VEs. Moreover, we applied the adapted/extended methods to both streamline the usability engineering process as well as provide sufficient coverage of the usability space. While the name implies that the various methods are applied in sequence, there is considerable opportunity to iterate both within a particular method as well as among methods. It is important to note that all the pieces of this approach have been used for years in GUI usability evaluations. The unique contribution of the Gabbard, Hix & Swan [1999] work is the breadth and depth offered by progressive use of these techniques, adapted when necessary for VE evaluation, in an application-specific context. Further, the way in which each step in the progression informs the next step is an important finding, as discussed near the end of this section.

Figure 3 presents the sequential evaluation approach. It allows developers to improve a VE's user interface by a combination of expert-based and user-based techniques. This approach is based on sequentially performing user task analysis (see figure 3, area labeled 1), heuristic (or

guidelines-based expert) evaluation (figure 3, 2), formative evaluation (figure 3, 3), and summative evaluations (figure 3, 4), with iteration as appropriate within and among each type of evaluation. This approach leverages the results of each individual method by systematically defining and refining the VE user interface in a cost-effective progression.

Depending upon the nature of the application, this sequential evaluation approach may be applied in a strictly serial approach (as figure 3's solid black arrows illustrate) or iteratively applied (either as a whole or per individual method as figure 3's white arrows illustrate) many times. For example, when used to evaluate a complex command and control battlefield visualization application [Hix et al., 1999], user task analysis was followed by significant iterative use of heuristic and formative evaluation, and lastly followed by a single, broad summative evaluation.

From experience, this sequential evaluation approach provides cost-effective assessment and refinement of usability *for a specific VE application*. Obviously, the exact cost and benefit of a particular evaluation effort depends largely on the application's complexity and maturity. In some cases, cost can be managed by performing quick and lightweight formative evaluations (which involve users and thus are typically the most time-consuming to plan and perform). Moreover, by using a "hallway methodology" [Nielsen, 1999], user-based methods can be performed quickly and cost-effectively by simply finding volunteers from within one's own organization. This approach should only be used as a last resort, or in cases where the representative user class includes just about anyone. When used, care should be taken to ensure that "hallway" users provide a close representative match to the application's ultimate end-users.

Each of the individual methods in the sequential evaluation approach is described in more detail below, with particular attention to how we (and others) have adapted them for VEs.

3.3.1 User Task Analysis

A *user task analysis* provides the basis for design in terms of what users need to be able to do with the VE application. This analysis generates (among other resources) a list of detailed task descriptions, sequences, and relationships, user work, and information flow (figure 3, A). Typically a user task analysis is provided by a VE design and development team, based on extensive input from representative users. Whenever possible, it is useful for an evaluator to participate in the user task analysis.

The user task analysis also shapes representative user task scenarios (figure 3, D) by defining, ordering, and ranking user tasks and task flow. The accuracy and completeness of a user task analysis directly affects the quality of the subsequent formative and summative evaluations, since these methods typically do not reveal usability problems associated with a specific interaction within the application unless it is included in the user task scenario (and is therefore performed by users during evaluation sessions). Similarly, in order to evaluate how well an application's interface supports high-level information gathering and processing, representative user task scenarios must include more than simply atomic, mechanical- or physical-level tasking, but should also include high-level cognitive, problem-solving tasking specific to the application domain. This is especially important in VEs, where user tasks generally are inherently more complex, difficult, and unusual than in, for example, many GUIs. Task analysis is perhaps the most over-looked phase of usability engineering, and is one of the most important, driving all subsequent activities in the usability engineering process.

3.3.2 Heuristic Evaluation

A *heuristic evaluation* or *guidelines-based expert evaluation* may be the first assessment of an interaction design based on the user task analysis and application of guidelines for VE user

interface design. One of the goals of heuristic evaluation is to simply identify usability problems in the design. Another important goal is to identify the usability problems *early in the development lifecycle* so that they may be addressed, and the redesign iteratively refined and evaluated [Nielsen & Mack, 1994]. In a heuristic evaluation, VE usability experts compare elements of the user interaction design to guidelines or heuristics (figure 3, B), looking for specific situations in which guidelines have been violated, and therefore are potential usability problems. The evaluation is performed by one or (preferably) more usability experts and does not require users. A set of usability guidelines or heuristics that are either general enough to apply to any VE or are tailored for a specific VE is also required.

Heuristic evaluation is extremely useful as it has the potential to identify many major and minor usability problems. Nielsen [1992] found that approximately 80 percent (between 74 percent and 87 percent) of a design's usability problems may be identified when three to five expert evaluators are used. Moreover, the probability of finding a given *major* usability problem may be as great as 71 percent when only three evaluators are used. From experience, heuristic evaluation of VE user interfaces provides similar results; however, the current lack of well-formed guidelines and heuristics for VE user interface design and evaluation make this approach more challenging for VEs.

Nonetheless, it is still a very cost-effective method for early assessment of VEs and helps uncover usability problems that, if not discovered via a heuristic evaluation, will very likely be discovered in the much more costly formative evaluation process. In fact, one of the strengths of the sequential evaluation approach is that usability problems identified during heuristic evaluations can be detected and corrected prior to performing formative evaluations. This approach creates a streamlined user interface design (figure 3, C) that may be more rigorously

studied in subsequent evaluations. Therefore, this approach leads to formative evaluation that is more cost-effective and efficient than a formative evaluation that is not based on a documented user task analysis and heuristic evaluation. In most cases, this approach avoids the situation where an iteration of formative evaluation is expended simply to expose obvious and glaring usability problems. A formative evaluation following a heuristic evaluation can focus not on the major usability issues, but rather on those more subtle and more difficult-to-recognize issues. This is especially important because of the cost of VE development.

Once both major and minor usability problems are identified, further assessment is needed to understand *how* particular interface components may affect user performance. To focus subsequent evaluations on these identified usability issues, evaluators use results of both the heuristic evaluation and the task analysis as the basis for representative user task scenarios (figure 3, D). For example, if heuristic evaluation identifies a possible mismatch between implementation of a voice recognition system and manipulation of user viewpoint, then scenarios requiring users to manipulate the viewpoint would be included in subsequent formative evaluations.

3.3.3 Formative Evaluation

Formative or *user-centered evaluation* [Scriven, 1967] is a type of evaluation that is applied during evolving or formative stages of design to ensure that the design meets its stated objectives and goals. Williges [1984] and Hix & Hartson [1993] extended formative evaluation to support evaluation of GUI user interfaces. The method relies heavily on usage context (e.g., user tasks, user classes, user motivation, etc.) as well as a solid understanding of human-computer interaction (and in the case of VEs, human-VE interaction). The purpose of formative evaluation is to iteratively assess and improve the usability of an evolving user interface design.

A typical formative evaluation cycle may begin with development of user task scenarios that are specifically designed to explore many facets of a user interface design. Task scenarios should provide ample coverage of tasks identified during a user task analysis. Representative users are recruited to work through the task scenarios as evaluators observe and collect data. Experienced usability evaluators follow a structured and scientific approach to data collection, resulting in large volumes of both qualitative and quantitative data. Both types of collected data are equally important parts of the formative evaluation process; quantitative data indicate that a user performance issue is present, qualitative data indicate where (and sometimes why) it occurred.

Collected data are analyzed to identify user interface components that both support and detract from user task performance and user satisfaction. Alternating between formative evaluation and (re)design efforts ultimately leads to an iteratively refined user interface design (figure 3, E). Refining the user interface design such that it efficiently and effectively supports all user tasks ensures that each comparison in a subsequent summative evaluation is fair (i.e., each design in the summative study is as good as it can possibly be in terms of usability).

3.3.4 Summative Evaluation

Summative or *comparative evaluation* is an assessment and statistical comparison of two or more configurations of user interface designs, user interface components, and/or ITs. Summative evaluation is generally performed after user interface designs (or components) are complete, and is a traditional factorial experimental design with multiple independent variables. Summative evaluation enables evaluators to measure and subsequently compare the productivity and cost benefits associated with different user interface designs. Comparing VE user interfaces requires a consistent set of user task scenarios (borrowed and/or refined from the

formative evaluation effort), resulting in primarily quantitative data results that compare (on a task by task basis) a design's support for specific user task performance.

A major impact of the formative to summative progression is that results from formative evaluations inform design of summative studies by helping determine appropriate usability characteristics to evaluate and compare in summative studies. There are invariably numerous alternatives that can be considered as factors in a summative evaluation. Formative evaluations typically point out the most important usability characteristics and issues (e.g., those that recur most frequently, those that have the largest impact on user performance and/or satisfaction, etc.). These issues then become strong candidates for inclusion in a summative evaluation. For example, if formative evaluation showed that users have a problem with format or placement of textual information in a heavily graphical display, a summative evaluation could explore alternative ways of presenting such textual information. As another example, if users (or developers) want a number of different display modes, such as stereoscopic and monoscopic, head-tracked and static, landscape view and overhead view of a map, these various configurations can also be the basis of rich comparative studies related to usability.

4 Comparison of approaches

The two major evaluation methods we have presented for VEs – testbed evaluation and sequential evaluation – take quite different approaches to the same problem, namely, how to improve usability in VE applications. At a high level, these approaches can be characterized in the space defined in section 3. Sequential evaluation is done in the context of a particular application and can have both quantitative and qualitative results. Testbed evaluation is done in a generic evaluation context, and usually seeks quantitative results. Both approaches employ users in evaluation.

In this section, we take a more detailed look at the similarities of and differences between these two approaches. We organize this comparison by answering several key questions about each of the methods. Many of these questions can be asked of other evaluation methods, and perhaps *should* be asked prior to designing a usability evaluation. Indeed, answers to these questions may help one identify appropriate evaluation methods given specific research, design, or development goals. Future work (by us and others) should attempt to find valid answers to these and other related questions regarding different usability evaluation methods. However, our immediate goal is to understand the general properties, strengths, and weaknesses of each approach so that the two approaches can be linked in complementary ways (see section 5).

4.1 What are the goals of the approach?

As mentioned above, both approaches ultimately aim to improve usability in VE applications. However, there are more specific goals that exhibit differences between the two approaches.

Testbed evaluation has the specific goal of finding generic performance characteristics for VE ITs. This means that we want to understand IT performance in a high-level, abstract way, not in the context of a particular VE application. This goal is important because if achieved, it can lead to wide applicability of the results. In order to do generic evaluation, the testbed approach is limited to general techniques for common, universal tasks (such as navigation, selection, or manipulation). To say this in another way, testbed evaluation is not designed to evaluate special-purpose techniques for specific tasks, such as applying a texture. Rather, it abstracts away from these specifics, using generic properties of the task, user, environment, and system.

Sequential evaluation's immediate goal is to iterate towards a better user interface for a particular application, in this case, a specific VE application. It looks very closely at particular user tasks of an application to determine which scenarios and ITs should be incorporated. In

general, this approach tends to be quite specific, to produce the best possible interface design for a particular application under development.

4.2 When should the approach be used?

By its non-application-specific nature, the testbed approach actually falls completely outside the design cycle of a particular application. Ideally, testbed evaluation should be completed before an application is even a glimmer in the eye of a developer. Since it produces general performance/usability results for ITs, these results can be used as a starting point for the design of new VE applications.

On the other hand, sequential evaluation should be used early and continually throughout the design cycle of a VE application. User task analysis is necessary before the first interface prototypes are built. Heuristic and formative evaluations of a prototype produce recommendations that can be applied to subsequent design iterations. Summative evaluations of different design possibilities can be done when the choice of design (e.g., for ITs) is not clear.

The distinct time periods in which testbed evaluation and sequential evaluation are employed suggests that combining the two approaches is possible, and even desirable. Testbed evaluation can first produce a set of general results and guidelines that can serve as an advanced and well-informed starting point for a VE application's user interface design. Sequential evaluation can then refine that initial design in a more application-specific fashion. We expand on this idea in section 5.

4.3 In what situations is the approach useful?

Testbed evaluation allows the researcher to understand detailed performance characteristics of common ITs, especially user performance. It provides a wide range of performance data that may

be applicable to a variety of situations. In a development effort that requires a suite of applications with common ITs and interface elements, testbed evaluation could provide a quantitative basis for choosing them, because developers could choose ITs that performed well across the range of tasks, environments, and users in the applications; their choices are supported by empirical evidence.

As we have said, the sequential evaluation approach should be used throughout the design cycle of a VE application, but it is especially useful in the early stages of interface design. Because sequential evaluation produces results even on very low-fidelity prototypes or design specifications, a VE application's user interface can be refined much earlier, resulting in greater cost savings. Also, the earlier this approach is used in development, the more time remains for producing design iterations, which ultimately results in a better product. This approach also makes the most sense when a user task analysis has been performed. This analysis will suggest task scenarios that make evaluation more meaningful and effective.

4.4 What are the costs of using the approach?

The testbed evaluation approach can be seen as very costly, and is definitely not appropriate for every situation. In certain scenarios, however, its benefits (see section 4.5) can make the extra effort worthwhile. Some of the most important costs associated with testbed evaluation include: difficult experimental design (many independent and dependent variables, where some of the combinations of variables are not testable), experiments requiring large numbers of trials to ensure significant results, and large amounts of time spent running experiments because of the number of subjects and trials. Once an experiment has been conducted, the results may not be as detailed as some developers would like. Since testbed evaluation looks at generic VE situations,

information on specific interface details such as labeling, the shape of icons, and so on will not usually be available.

In general, the sequential evaluation approach may be less costly than testbed evaluation because it can focus on a particular VE application rather than paying the cost of abstraction. However, some important costs are still associated with this method. Multiple evaluators may be needed. Development of useful task scenarios may take a large amount of effort. Conducting the evaluations themselves may be costly in terms of time, depending on the complexity of task scenarios. Most importantly, since this is part of an iterative design effort, time spent by developers to incorporate suggested design changes after each round of evaluation must be considered.

4.5 What are the benefits of using the approach?

Since testbed evaluation is so costly, its benefits must be significant before it becomes a useful evaluation method. One such benefit is generality of the results. Since testbed experiments are conducted in a generalized context, the results may be applied many times in many different types of applications. Of course, there is a cost associated with each use of the results, since the developer must decide which results are relevant to a specific VE. Secondly, testbeds for a particular task may be used multiple times. When a new IT is proposed, that technique can be run through the testbed and compared with techniques already evaluated. The same set of subjects is not necessary since testbed evaluation usually uses a between-subjects design. Finally, the generality of the experiments lends itself to development of general guidelines and heuristics. It is more difficult to generalize from experience with a single application.

For a particular application, the sequential evaluation approach can be very beneficial. Although it does not produce reusable results or general principles in the same broad sense as

testbed evaluation, it is likely to produce a more refined and usable VE than if the results of testbed evaluation were applied alone. Another of the major benefits of this method relates to its involvement of users in the development process. Since members of the representative user group take part in many of the evaluations, the VE is more likely to be tailored to their needs, and should result in higher user acceptance and productivity, reduced user errors, increased user satisfaction, and so on. There may be some reuse of results, because other applications may have similar tasks or requirements, or may be able to use refined ITs produced by the process.

4.6 How are the approach's evaluation results applied?

The results of testbed evaluation are applicable to any VE that uses the tasks studied with a testbed. Currently, testbed results are available for some of the most common tasks in VEs: travel and selection/manipulation [Bowman, Johnson, & Hodges, 1999]. The results can be applied in two ways. The first, informal, technique is to use the guidelines produced by testbed evaluation in choosing ITs for an application (as in [Bowman, Johnson, & Hodges, 1999]). A more formal technique uses the requirements of the application (specified in terms of the testbed's performance metrics) to choose the IT closest to those requirements. Both of these approaches should produce a set of ITs for the application that makes it more usable than the same application designed using intuition alone. However, since the results are so general, the VE will almost certainly require further refinement.

Application of results of the sequential evaluation approach is much more straightforward. Heuristic and formative evaluations produce specific suggestions for changes to the application's user interface or ITs. The result of summative evaluation is an interface or set of ITs that performs the best or is the most usable in a comparative study. In any case, results of the evaluation are tied directly to changes in the interface of the VE application.

5 Links between testbed and sequential evaluation

Based on this analysis of the testbed evaluation and sequential evaluation approaches to VE evaluation, we have found that there are many ways in which these approaches can influence and affect one another when used together as part of a broader approach. To this end, we have identified a number of ways that the results of one approach can be used to strengthen and refine the other.

As we have noted, there is an inherent separation between the two approaches. Although both have the eventual goal of improving the usability of VE applications, testbed evaluation does this indirectly, through evaluation in a generic context, while the sequential evaluation approach assesses applications directly. However, this does not mean that the two processes are mutually exclusive, or that they are incompatible. On the contrary, we have found many ways the two approaches can influence and benefit one another, and even situations in which they can be used together.

5.1 Testbed evaluation as input to sequential evaluation

There are several ways in which testbed evaluation can affect the sequential evaluation approach (see Figure 4 for a summary). User task analysis, a critical part of the sequential evaluation approach, requires an understanding of tasks users must perform and possible ITs that could be used to accomplish those tasks. Taxonomic structures from the testbed approach provide both of these. Taxonomies provide a standard way to organize and decompose a task, and they contain a design space from which many ITs can be built.

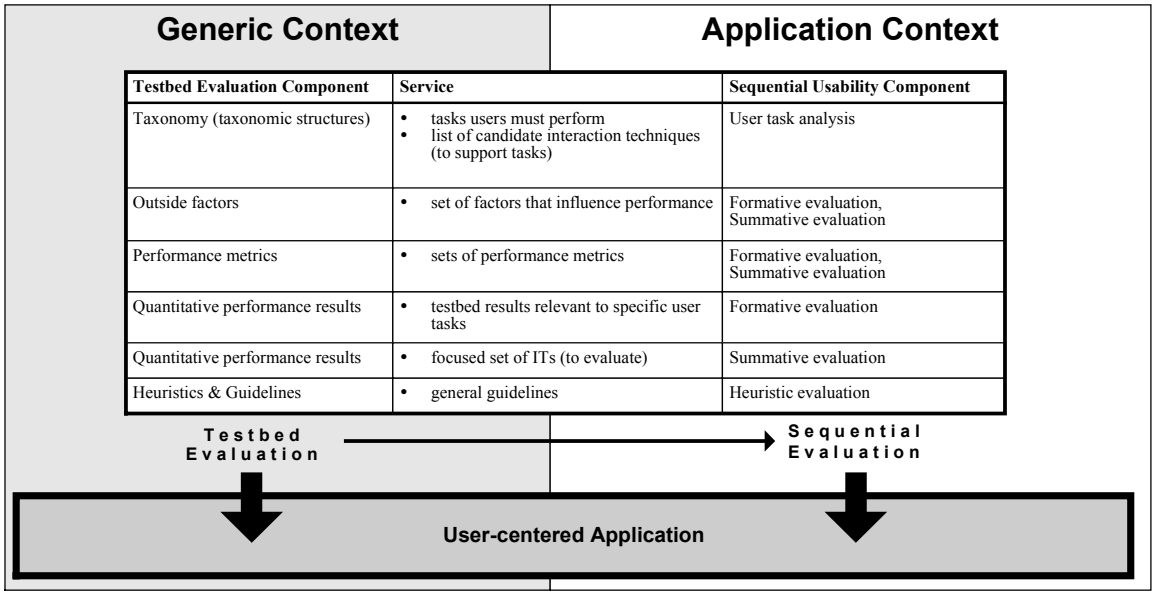

Figure 4. Testbed evaluation as input to sequential evaluation.

The set of factors other than ITs that could influence performance (outside factors) are an important component of the testbed evaluation process, since they are candidates for independent variables in testbed experiments. For example, one could test whether the number of obstacles in an environment affects the speed of traversing a path in that environment. These same factors can play a role in shaping formative and summative evaluation components of the sequential evaluation approach. The evaluator can use these factors to more carefully plan task scenarios that assess the range of potential interactions a user could have with the VE. In a similar way, sets of performance metrics defined for testbed evaluation are useful in formative and summative evaluation. These metrics can be checked to ensure that the evaluator observes all variables that contribute to a usable interface.

Quantitative performance results obtained from testbed experiments can play a role in the sequential evaluation process. In formative evaluation, an evaluator is trying to produce one or more usable ITs that can later be compared. If testbed results are available for the task in question, incorporation of these ITs into a VE can begin at a much more refined level based on

performance results. In the same way, testbed results can help narrow the set of ITs in summative evaluation. The relative performance of two ITs may already be known through testbed evaluation, or a particular IT may be known to perform badly in the situation presented by a particular VE application. In any case, these results should be considered before beginning either type of evaluation.

Finally, the general guidelines produced by testbed evaluation can serve as input for heuristic evaluation in the sequential evaluation approach. In fact, this addresses a potential problem with using heuristic evaluation for VEs: a lack of VE-specific heuristics. Since guidelines from the testbed approach are based on experimental evidence, heuristic evaluation using these guidelines should produce a more usable initial design to be fed to the formative evaluation process.

5.2 Sequential evaluation as input to testbed evaluation

Linking of these two approaches can also proceed in the opposite direction, with the sequential evaluation approach serving to inform and refine testbed evaluation. We suggest three ways this might take place. In all three of these cases, the experiences of analyzing a real-world application help to refine the generic model used for testbed evaluation (see Figure 5 for summary).

One way this can occur involves the process of user task analysis. Task analysis takes place in the context of a particular application, and can also be refined as the sequential evaluation approach is iterated. This can result in a quite detailed understanding of user tasks, intentions, and mental models for a specific VE. This understanding is exactly what is needed to create good taxonomies of ITs for a particular task, since taxonomies in the testbed approach are based on task decomposition. If taxonomies more closely fit the user's model of a particular task, when

this taxonomy is used as a framework for evaluation the results should be a better predictor of user performance in real systems.

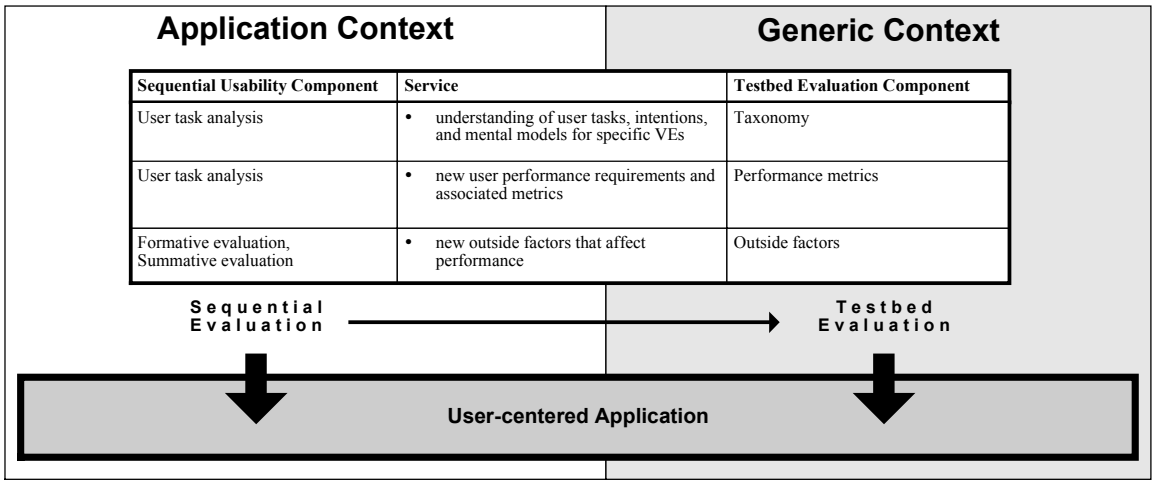

Figure 5. Sequential evaluation as input to testbed evaluation.

Subsequent to the process of user task analysis, usability goals and associated metrics can be determined. It is important for a user to complete tasks efficiently, correctly, without frustration, and in comfort. These characteristics match some of the possible performance metrics given by the testbed approach. However, it is possible that in the process of user task analysis and subsequent setting of usability goals, evaluators will find that a VE has a requirement whose fulfillment cannot be determined using any of the listed performance measures. The requirement may suggest a new metric to be added to the list and included in future testbed experiments.

It is difficult in the testbed approach to come up with complete lists of the outside factors that could affect user performance. This is often done based on the evaluators' intuition alone. However, experiences of evaluators performing formative and summative evaluations can add to and refine these lists. Evaluators may notice that a user performing a particular task is greatly affected by some characteristic of the environment. This would suggest that this characteristic should be studied in a future testbed experiment to determine the extent of its effects more

generally. If that variable has already been studied in a general experiment, it may be possible to give more weight to this factor in analysis of the results.

5.3 Usage scenarios

While the links between approaches described above appear to provide rich coverage of the usability space, we recognize that it is likely too complex and time-consuming to apply all of them to a single VE development effort. Nonetheless, there are research and development arrangements that are well-suited for this integrated approach, including development of a suite of VE applications as well as distributed, asynchronous research and development.

It is reasonable to assume that as VE hardware and user interfaces become more accessible to the mainstream public, there will be significant interest in developing "productivity tools", or software applications that allow users to perform real work, for extended periods, within a VE. Thus, it can be expected that suites of software applications may be developed that resemble, for example, the Microsoft™ Office suite of tools. In this case, early research and development of common user interface components and ITs could be furthered by those usability evaluation methods that evaluate in a generic context (such as the testbed evaluation approach). During later stages of research and development, specific applications within the suite could be evaluated using the application-specific evaluation methods (such as the sequential evaluation approach).

But perhaps the most likely environment in which the links may be applied is a distributed, asynchronous research setting. In this case, researchers performing generic evaluation of ITs, input/output devices, and user interface components can provide insight, recommendations, and guidelines to the community at large. Subsequently, those performing evaluation of specific applications may use results published from the generic evaluation efforts to aid in their specific application evaluation effort. As described in sections 5.1 and 5.2, the fact that each type of

evaluation effort may aid the other introduces the possibility for powerful collaboration among researchers interested in usability evaluation of VEs.

6 Conclusions and future work

Clearly, performing usability evaluation on non-traditional interactive systems requires new approaches, techniques, and insights. While VE evaluation at its highest level retains the same goals and conceptual foundation as its GUI predecessors, the practical matter of performing actual evaluations can be quite different. We have shown that this is especially true for VEs, and have outlined some of the distinctive characteristics of VE evaluation as well as several possible approaches. This information alone is practical to VE developers and researchers in producing usable applications.

The links described in section 5 combine approaches that produce quantitative results with those that produce qualitative results, those that evaluate in a generic context with those that evaluate specific applications, and those that require users with those that do not. By considering all these approaches, evaluators can converge more quickly on a usable system. As we have detailed, each approach brings with it certain advantages that are synergistic when multiple approaches are used.

We plan to continue this work on several fronts. First, we will continue to evaluate real-world VE systems for usability, using the combined approach we describe here. This should lead to a greater understanding of the practical process that can be used to perform evaluation more efficiently and with better results. Second, there are certain VE interaction tasks that have not been explored sufficiently. For example, the task of VE system control, in which the user wishes to issue a command or change the state of the system in some way, is not well-understood. Generic evaluations of various system control techniques would be highly useful to the VE

community. Third, we hope others will join us in analyzing usability evaluation methods in terms of the questions posed in section 4. Answers to these and similar questions, for a broader variety of evaluation approaches, can greatly increase the effectiveness and efficiency of performing such evaluations. Such results could help expand the breadth and depth of usability evaluations performed on VE user interfaces. Finally, it is a reality that many VE developers do not choose to perform full usability studies on their systems, making the availability of useful and practical guidelines for VE interface design invaluable. We plan to use our extensive experience in VE usability evaluation to create and integrate sets of such guidelines that can be disseminated widely among developers.

Acknowledgments

Portions of this research were funded by the Office of Naval Research, Dr. Helen M. Gigley, Program Manager. Dr. Gigley has funded an on-going collaboration between Virginia Tech and the Naval Research Laboratory in Washington, D.C. for several years. Dr. Ed Swan, of the Naval Research Laboratory, has been a close collaborator on much of this work, also supported by Dr. Larry Rosenblum of NRL. Dr. Paul Quinn and Dr. Astrid Schmidt-Nielsen, also of the Office of Naval Research, have recently provided funding for our efforts. Dr. Richard E. Nance, of Virginia Tech's Systems Research Center, has given much moral support to our research. Dr. Larry F. Hodges of Georgia Tech was instrumental in research on testbed evaluation. We would also like to thank Donald Johnson, Don Allison, and Drew Kessler for their help and support. We are grateful to all these contributors, without whom this large body of work would not have been possible.

References

Bowman, D. (2001). Principles for the Design of Performance-Oriented Interaction Techniques. To appear in Stanney, K. (Ed.). *Handbook of Virtual Environment Technology*, Lawrence Erlbaum Associates.

Bowman, D. and Hodges, L. (1999). Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. *The Journal of Visual Languages and Computing*, *10*(1), 37-53.

Bowman, D., Johnson, D., and Hodges, L. (1999). Testbed Evaluation of VE Interaction Techniques. Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 26-33.

Card, S., Mackinlay, J., & Robertson, G. (1990). The Design Space of Input Devices. Proceedings of CHI: Human Factors in Computing Systems, 117-124.

Darken, R. P. and Sibert, J. L. (1996). Wayfinding Strategies and Behaviors in Large Virtual Worlds. In *Proceedings of CHI '96: ACM conference on Human Factors in Computing Systems*, 142-149.

Gabbard J. L. (1997). A Taxonomy of Usability Characteristics for Virtual Environments. Masters Thesis. Department of Computer Science, Virginia Tech.

Gabbard, J. L., Hix, D, and Swan, E. J. (1999). User Centered Design and Evaluation of Virtual Environments , *IEEE Computer Graphics and Applications*, *19*(6), 51-59.

Hix, D., Swan, E. J., Gabbard, J. L., McGee, M., Durbin, J., and King, T. (1999). User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment. In *Proceedings of IEEE Virtual Reality '99*, 96-103.

Hix, D. and Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability through Product & Process.* New York, John Wiley and Sons.

Kaur, K. (1998). Designing virtual environments for usability. PhD thesis. Centre for HCI Design, City University, London.

Kennedy, R.S., Lane, N.E., Berbaum, K.S., and Lilienthal, M.G. (1993). Simulator sickness questionnaire (SSQ): A new method for quantifying simulator sickness. *International Journal of Aviation Psychology*, *3*, 203-220.

Kennedy, R.S., Stanney, K., and Dunlap, W. (2000). Duration and Exposure to Virtual Environments: Sickness Curves During and Across Sessions. *PRESENCE: Teleoperators and Virtual Environments*, *9*(5), 463-472.

Nielsen, J. (1999). Users First: Cheap Usability Tests. Available at: http://www.zdnet.com/devhead/stories/articles/0,4413,2224316,00.html.

Nielsen, J. and Mack, R. L. (1994). Executive summary. In Nielsen, J. & Mack, R. L. (Ed.), *Usability Inspection Methods*, New York, John Wiley & Sons, 1-23.

Polson, P., Lewis, C., Rieman, J., and Wharton, C. (1992). Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *International Journal of Man-Machine Studies*, *36*, 741-773.

Poupyrev, I., Weghorst, S., Billinghurst, M. and Ichikawa, T. (1997). A Framework and Testbed for Studying Manipulation Techniques for Immersive VR. In *Proceedings of VRST '97: ACM Symposium on Virtual Reality Software and Technology*, 21-28.

Scriven, M. (1967). The methodology of evaluation. In R. E. Stake (Ed.), *Perspectives of curriculum evaluation*, American Educational Research Association Monograph. Chicago, Rand McNally.

Siochi, A. C. and Hix, D. (1991). A Study of Computer-Supported User Interface Evaluation Using Maximal Repeating Pattern Analysis. In *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*.

Slater, M., Usoh, M., and Steed, A. (1995). Taking Steps: The Influence of a Walking Metaphor on Presence in Virtual Reality. *ACM Transactions on Computer Human Interaction*, *2*(3) 201-219.

Stanney, K. (1999). Personal communication.

Steed, A. and Tromp, J. (1998). Experiences with the Evaluation of CVE Applications. In *Proceedings of Collaborative Virtual Environments*.

Williges, R. C. (1984). Evaluating Human-Computer Software Interfaces. In *Proceedings of International Conference on Occupational Ergonomics*.

Witmer, B. G. and Singer, M. J. (1998). Measuring Presence in Virtual Environments: A Presence Questionnaire. *PRESENCE: Teleoperators and Virtual Environments*, *7*(3), 225-240.