ABSTRACT
Usability evaluation of an information visualization technique can only be done by the joint evaluation of both the visual representation and the interaction techniques. This work proposes task models as a key element for carrying out such evaluations in a structured way. We base our work on a taxonomy abstracting from rendering functions supported by information visualization techniques. CTTE is used to model these abstract visual tasks as well as to generate scenarios from this model for evaluation purposes. We conclude that the use of task models allows generating test scenarios which are more effective than informal and unstructured evaluations.

Author Keywords
Visualisation tasks, task modeling, information visualization, evaluation.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces. Graphical user interfaces (GUI), Evaluation/methodology.

INTRODUCTION
In the last few years the increasing volume of information provided by several applications, different instruments and mainly the Web has lead to the development of techniques for selecting among a bulk of data the subset of information that is relevant for a particular goal or need. Research on scientific data visualization, visual query systems, data mining and interactive visualization techniques has resulted in a wide variety of visual presentation and interaction techniques that can be applied in different situations.

However, although there is a great variety of models and techniques for information visualization (see Card et al. [2]), each application requires a particular study in order to determine if the selected technique is useful and usable. The type of data that should be represented as well as the user tasks or analysis process that the visualization should help or support usually guides these studies.

Potential users of information visualization systems often have their own analysis tools (statistical ones, for example) and are not aware of the benefits of visualization techniques as a first phase in the data analysis process. Providing ways to measure the effectiveness of these visualization techniques under different scenarios can improve not only the development of such techniques but the application domain areas as well.

Whereas the first information visualization techniques were presented without thorough evaluation studies, recently, researchers have been more aware of the importance of such usability studies [3]. Almost all the evaluations are accomplished through user testing which lead us to the problem of defining the set of user tasks that should be part of the experiments, as well as providing different datasets to be tested. Moreover, the evaluation of information visualization techniques should be based on empirical testing of both the visual representation and the interaction mechanisms. For example, usual and critical aspects of visual representations are object occlusion and visual disorder, while visual disorientation is caused by changes in the visual representations due to some user action. Thus, often there are situations when one aspect (interaction) affects the other (visual representation). All such characteristics should be verified in order to evaluate a specific visualization technique.

The tasks the user performs to support the data analysis and comprehension processes are well known. However, few authors explicitly explore the set of user tasks in order to guarantee that experiments applied for evaluation purposes cover all the possible scenarios a user might face [11]. In addition, there are no works in the domain of information visualization discussing how tasks models can improve the design of tests based on scenarios.

Our work tries to close this gap by investigating the use of task models built with CTTE notation [12] to model common tasks supported by information visualization techniques, as well as the generation of scenarios from this model for evaluation purposes. We aim at proving that when using formal methods for task modeling the set of test scenarios we can generate is much more effective
in a variety of situations (e.g. comparison of different techniques, effect of different interaction devices, isolation of aesthetics and interaction issues, etc.).

The paper is organized as follows. Next section discusses user tasks in information visualization techniques, and presents the classification of visualization tasks we adopted. Then, we briefly describe CTT so the task modeling we present in the sequence can be easily comprehended. Finally, we present a study on how two well-known techniques (Treemaps [8] and hyperbolic browser Magnifind [9]) implement some abstract visual tasks. We discuss our results in the last section.

TASKS IN INFORMATION VISUALIZATION TECHNIQUES

Usually, information visualization techniques provide a set of operations or interactive (visual) techniques that allow a user to accomplish specific-domain tasks. Weherend and Lewis [16] and Springmeyer [14] in the early 90’s were among the first ones to explicitly address user operations and tasks characterizing the data analysis process in order to facilitate the selection of adequate visual representations. Both sets of tasks are domain-independent, which allows generalizing their classification.

Weherend and Lewis [16] classified operations that a user might need to execute to analyze data as:

- **Locate**: the user knows a dataset entry and indicate it by pointing or describing it.
- **Identify**: similar to locate but the user describe the dataset entry without knowing it previously.
- **Distinguish**: different objects should be presented as distinct visual items.
- **Categorize**: objects may be different because they belong to different categories, which should be described by the user.
- **Cluster**: the system may find out categories and objects belonging to them are shown linked or grouped together.
- **Distribution**: the user specifies categories and objects belonging to them are distributed among them.
- **Rank**: the user is asked to indicate the order of the objects displayed.
- **Compare**: the user is asked to compare entities based on their attributes.
- **Compare within and between relations**: the user is asked to compare similar entities or different sets of objects.
- **Associate**: the user is asked to establish relations between objects displayed.
- **Correlate**: the user may observe shared attributes between objects.

In that work, through the classification of typical visual representations based on an integrated analysis of these categorization and the characteristics of datasets, one could find out which visual representation would lead to a better solution for an application problem. Later on, Zhou and Feiner [18] introduced another categorization of tasks. They separated presentation intents (goals a user has when using a visual representation) from low-level visual techniques (the exact operation performed on a given object presented in the display) by means of an intermediate level, the visual tasks. Visual tasks can be considered abstracted visual techniques, since they indicate a desired visual effect in the representation while a visual technique is a way to achieve that desired effect, either by the user or the system.

Zhou and Feiner characterize visual tasks along two dimensions: visual accomplishments and visual implications. Visual accomplishments correspond to the presentation intents a visual task is supposed to support while visual implications specify the visual techniques that could be used to fulfill the visual task. Regarding visual accomplishments, two classes of visual tasks can be identified: inform and enable. Inform tasks can be further distinguished as Elaborate and Summarize tasks, while enable tasks can be divided in explore tasks and compute tasks. At the bottom level of this hierarchy of abstract visual tasks one still has tasks like categorize, cluster, compare, correlate, identify, etc., i.e., generic operations like those identified by Weherend and Lewis [16]. For example, one of the explore tasks is the abstract search, which can be represented by the visual tasks categorize, cluster, compare, correlate, distinguish, emphasize, identify, locate, rank, and reveal [18]. Based on the observation of principles for visual perception and cognition, the same authors [18] also establish a link from the visual tasks to the adequate visual techniques. For example, to identify a piece of information, one can give its name, point at it in the display, give a range of attributes as a profile, all of these implying certain concrete visual tasks like name input, mouse pointing, filtering.

Morse et al. [11] developed a (manual) procedure for mapping from this visual taxonomy to concrete tasks represented by 50 questions in the information retrieval domain. They used subsets of these questions and simple visual prototypes to test the role of visualizations in that domain. By defining tests based on this taxonomy, they tried to exhaustively evaluate capabilities of visualizations.

In the next sections we investigate how to model these tasks using a formal method and its corresponding environment to take advantage of the possibilities of automatically generating different scenarios that cover all the accomplishments and implications for each visual task.

AN APPROACH FOR SCENARIO-BASED EVALUATION OF INFORMATION VISUALIZATION TECHNIQUES

Basic principles

Ideally, the evaluation of visualization techniques must be able to:

- Identify the user goals and verify if the user can reach them with an application which implements an information visualization technique;
Identify which interaction mechanisms made available to the user by the visualization techniques are useful to accomplish the user task;

Identify the graphical rendering functions that have been employed by the visualization techniques to show information;

Relate user goals, interaction mechanisms and graphical rendering.

Informal scenarios currently employed in usability testing of visualization techniques fail to cope with the requirements stated above because they just provide imprecise descriptions, often ambiguous, of what must be done and which aspects must be covered during the evaluation.

The solution we propose hereafter to better explore scenarios during evaluations is to extract them from task models rather than use ad-hoc informal scenarios.

**Modeling process**

Tasks concerning the use of information visualization systems must be considered at different abstraction levels as presented in Table 1. At the highest level (more abstract) there is the user goal and intents. The so-called abstract level covers a set of generic subtasks which are required to accomplish the user goal. The third level, the interaction level, includes the specification of the activity performed by the user, the system or both (interactive tasks). The lowest level, called visual presentation, includes a concrete description of the system activity in terms of rendering function employed to provide user with information concerning the data.

<table>
<thead>
<tr>
<th>Task level</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>user goals and intents</td>
</tr>
<tr>
<td>Abstract</td>
<td>generic tasks</td>
</tr>
<tr>
<td>Interaction</td>
<td>user and system activity</td>
</tr>
<tr>
<td>Visual presentation</td>
<td>rendering functions performed by the application</td>
</tr>
</tbody>
</table>

Table 1. Task abstraction level and scope.

Rendering functions would be considered as tasks by themselves since they do not have any specific user goal associated to them. They just describe a particular strategy employed by the application to provide users with information. However, we decided to include these rendering functions in our modeling in order to show where they appear during the execution of a particular task.

Rendering functions comprehend the layer covered by the visualization technique and it can be used to differentiate one particular visualization technique from other suitable to represent the same kind of data.

Several rendering functions can be combined in a single visualization technique. Moreover, different rendering functions might equally satisfy user requirements for visualizing data or be combined to present data. Thus, rendering functions are often optional or associated to operators such as choice and concurrency.

**Scenario extraction from task models**

Following our approach, the extraction of scenarios is a straightforward process which requires the following steps:

1) Produce unambiguous representations of user goal and intents as well as the logical decomposition of subtasks required to accomplish that goal;

2) Define the rendering functions (how the application will be presented in the tasks) that must be checked during the evaluation;

3) Associate rendering functions to interaction mechanisms enabling the visualization;

4) Extract a set of possible scenarios.

The first step typical is covered by task models. The second step relies on previously defined visualization techniques that just need to be integrated in the task model. At this point one must consider rendering functions as abstractions of graphical representations in visualization techniques. Such visualization techniques are well known in the information domain [14][16][18]. This step is better described in the next section. Step 3 creates relationships between rendering functions and task in the task model. Then, step 4 is accomplished by playing the task model in order to check all possible sequences for the task. Each individual sequence is considered one possible scenario to the application.

**Implementing the evaluation**

The evaluation is done by playing these scenarios in the application which implements the visualization technique. Theses scenarios are used to instrument an evaluation based on the inspection of the interface.

Besides the testing of usability of a specific visualization technique, such scenarios can be used to compare different techniques. This alternative was selected as our case study described further in the paper. By playing scenarios with different information visualization techniques, evaluations can identify which rendering functions and interaction mechanisms are available in each technique. Moreover, evaluators would point out required functionalities that are missing in the visualization techniques.

**Describing visualization tasks with a notation of task models**

In this section we employ the CTT notation [12] to present our tasks models. CTT has been chosen because it enables us to explicitly represent the tasks that are performed by the user, by the application and those are interactive (require both system and user intervention).

**CTT Basics**

ConcurTaskTree (CTT) notation [12] allows modeling 4 types of tasks: abstract task, user tasks, application task and interactive task (Figure 1). The Abstract tasks in CTT...
are tasks which require complex actions. User tasks are entirely performed by the user without interacting with the system, while Interactive tasks are performed by the user with the system. Application tasks describe actions performed by the system without user intervention.

![Abstract task User task Application task Interactive task]

**Figure 1. Symbols used to represent different tasks in CTT.**

In CTT, task models are organized in a hierarchical structure built according to a set of possible relationships between tasks, which are based on LOTOS operators such as choice (operator []), enabling (operator ![]), enabling with information passing (operator []![]), task interruption (operation >), etc. A set of tools called CTT Environment (CTTE [10]) was developed to edit, simulate and analyze CTT models. CTTE was used in our modeling.

**Modeling Visual Presentation with CTT**

When employing information visualization in data analysis, users perform several tasks. Such tasks have been classified by Zhou and Feiner [18] who separate the presentation intents (goals a user has when using the visual representation) from low-level visual techniques by means of an intermediate level, the so-called visual tasks. Visual tasks correspond to a set of patterns employed by several information visualization techniques. These patterns are modeled as generic rendering functions describing part of the system’s behavior during the execution of real user tasks. Thus, this approach enables us to reuse a set of well-defined visual tasks to evaluate visualization techniques simply changing rendering functions.

![Figure 2. Identify task modeled with CTT.]

Most of the visual tasks correspond (but not only) to the rendering functions. Figure 2 exemplifies this principle with a cognitive user task identify. Users can employ different strategies to perform this task such as to identify the searched object by its name (a textual attribute), portray it (i.e., using a graphic image) or giving a profile (i.e., set of attributes the searched object has).

The visual tasks identify, emphasize, reveal and cluster (see Figures 3, 4, and 5, respectively) are good examples of rendering functions. The modeling of rendering functions with CTT notation might require some particular adaptations. They are represented in our modeling as CTT application tasks which might be a minor limitation of the CTT notation since rendering functions are not used to be represented as part of a task model. In fact, they are better described by automata (or other kind of formalism allowing the description of system models). The reason why we force the representation of rendering function as application tasks is that this allows us to be declarative in our approach rather than procedural. This means we can include in the task model what is available, rather than stating how it is made available. This is the reason why on our models we only represent that various visualization techniques are available (by means of rendering functions) and they are optional and can be used in any order. The practical advantage of this is that we can produce scenarios which are adapted to evaluation purposes.

Figure 3 presents the so-called rendering function reveal. Several strategies can be employed to produce the expected result such as Expose, Itemize, Specify and Separate the object to be revealed by the application. These rendering functions can appear isolated or combined in a visualization technique. In order to represent this, they are connected by the symbol || indicating the concurrency between them. Moreover, since some visualization techniques may not implement all of these strategies, they are presented between brackets ([ ]) meaning they are optional. The symbol > associated to the abstract task reveal indicates that these tasks may be interrupted at any moment by an external event produced by the user.

![Figure 3. Reveal modeled with CTT.]

Figure 4 presents the modeling for the rendering function cluster, which consists of outline and individualize. The meaning of each strategy is given by the context where they are called. Differently of reveal, only one sub-strategy (outline or individualize) can be called at a time and both are optional in the modeling.

![Figure 4. Cluster task modeled with CTT.]

Each rendering function can be further decomposed to include low-level visual techniques which correspond to specific operations performed on a given object presented in the display. This is exemplified in Figure 5 which shows the modeling for emphasize. Structurally, emphasize is very similar to the task reveal (see Figure 5a). It consists of a set of optional and concurrent
rendering functions (Focus, Isolate and reinforce). Figure 5b show how a particular strategy (Reinforce) has been further decomposed to include specific implementation constraints. For example, the rendering functions could be achieved either by highlighting the colors or zooming over graphic elements.

![Diagram](image1)

**Figure 5.** Emphasize tasks modeled with CTT.

**Describing real user tasks**

It should be noticed that the patterns for visualization techniques described in the previous section, the so-called visual tasks, do not directly address user tasks. However, they can be employed to describe part of real user tasks. They can be combined with other tasks to produce models that describe/model possible user activity.

Let us consider a simple user task such as to locate a file in a file system, as presented in Figure 6. If users are not able to identify the file while in the root directory they will need to explore the file system to find it. Users may also finish the task anytime.

The explore task is presented as a complex task which requires users to select a folder (an interactive task) or request the application to reorganize the files in some way (according to a particular visualization technique).

The strategies that can be employed by the application to reorganize the information space are presented in the model as a combination of the rendering functions emphasize, reveal and cluster.

The abstract search task groups all possible strategies to locate a file. The symbol * associated to this task inform that it can be performed several times. Each iteration in the task explore requires users to select a folder, then the system shows the new file system configuration so that the user can evaluate whether s/he has found the file. At anytime the user can stop the searching by following the task finish. The interruption of a task is represented by the operator >.

When comparing the modeling presented in Figure 6 with the abstraction level proposed in Table 1, we can observe that the task locate file corresponds to the level goal. The sequence of tasks identify (file at root directory), explore and finish the application (anytime) corresponds to the generic task modeling covered by the abstract level. The level visual representation corresponds to the rendering functions (emphasize, reveal and cluster); notice that these rendering functions have been smoothly introduced in the task modeling. The level interaction is covered by the other tasks between the level abstract and visual representation.

**CASE STUDY**

In order to exemplify our approach to support usability evaluation, this section presents a short case study using two well known visualization techniques to be compared under the different scenarios extracted from the task locate file presented in Figure 6.

**Comparing visualization techniques**

We have selected two techniques, Treemaps [8] and the Hyperbolic browser [9], which are presented in figures 7 and 8, respectively.

![Diagram](image2)

**Figure 6.** CTT model of user task “locate a file”.

169
Treemaps [8] provides a visual representation of a hierarchical structure by recursively dividing a rectangular area, which represents the root, into rectangles (nodes). The size of the rectangles is proportional to some attribute (e.g. the size of files if the structure represents a file system). So, although it represents some important attributes, it does not communicate structure as well as traditional node-edge diagrams. The Treemaps version we used is Treemaps 3.2, which facilitates selection of information by filtering widgets.

The Hyperbolic browser [9] consists of a diagram represented using hyperbolic geometry and mapped to a unit disc in such a way that the size of a node is dependent of its distance from the disc center. This is a focus+context technique, where the whole hierarchy is shown in an integrated view. The selected node is in the disc center; the nodes are largest near the center, and become smaller towards the disc perimeter.

There are several tools that implement hyperbolic browser technique. The implementation we used in our case study was Magnifind.

Scenarios extracted
The simple model presented herein (Figure 6) implies several possible strategies to reach the goal (find a file) and thousands of different scenarios can be extracted from the task model. Due to space reasons we have limited our case study to the four scenarios which are presented in Figure 7.

Extracting scenarios from CTT models is eased by the tool CTTE which allows the execution of task models. Thus, the scenarios presented below are the direct result of the execution of a set of actions performed with CTTE.

In order to compare the two techniques we used the same file system when executing the proposed scenarios. The concrete task we choose to represent scenarios is “Locate file java.exe in the j2sdk1.4.2_0.4/bin folder”.

Playing scenarios to evaluate Treemaps
In Scenarios 1 and 2, a user has to identify a file by name selecting a single folder. The rendering function itemize is not implemented in Treemaps, while expose is.

Finding the file can be accomplished in two ways. Since the bin directory is at the second level, the user can find the file by moving the cursor on the area occupied by the bin directory until finding the desired file (Figure 7a) or the user can double-click on the bin directory and directly see the java.exe file (Figure 7b) since the whole area will be allocated to the files and directories belonging to bin. Actually we may say that the rendering function expose is implemented through two visual techniques: by showing the name attribute in a text box while the cursor moves on the display, and as the direct result of space subdivision when double-clicking in the directory.
Scenarios 3 and 4 can not be performed with Treemaps due to the rendering functions *outline* and *cluster* are not instantiated in this tool.

**SCENARIO 1** – File is identified after the first iteration, the visualization strategy is itemize

(application task) Task number 1: select folder

(user task) Task number 3: identify by name

(interactive task) Task number 4: finish

**SCENARIO 2** – File is identified after the first iteration, the visualization strategy is expose

(application task) Task number 1: select folder

(user task) Task number 3: identify by name

(interactive task) Task number 4: finish

**SCENARIO 3** – File is identified after two iterations, the visualization strategy is outline (see cluster)

(application task) Task number 2: Outline

(user task) Task number 3: not found

(interactive task) Task number 4: select folder

(application task) Task number 5: Outline

(user task) Task number 6: identify by name

(interactive task) Task number 7: finish

**SCENARIO 4** – File is identified using different strategies, n iterations (emphasize-focus) and itemize

(application task) Task number 2: Focus

(application task) Task number 4: select folder

(application task) Task number 5: Focus

(application task) Task number 7: Itemize

(user task) Task number 4: identify by name

(interactive task) Task number 5: finish

Figure 7. Four possible scenarios extracted from the task *locate a file*.

Playing scenarios to evaluate the Hyperbolic browser

Taking also Scenarios 1 and 2, we observe that the Hyperbolic browser cannot provide a way to locate the file directly because the bin directory is not displayed at first. Scenario 3 also cannot be performed because the rendering task *outline* is not implemented in Magnifind. However, Scenario 4 can be performed because the tool implements navigation allowing the user to successive focus nodes in the diagram, until reaching the desired node. The user navigates by manipulating the diagram in order to focuse first the j2sdk1.4.2_0.4, then the bin folder (Figure 8). At this point, in the Magnifind implementation, the selection of the bin folder by double-clicking it brings the Explorer window so the user has a list (itemized) view of the directory and can identify the file by name.

**DISCUSSION AND CONCLUSIONS**

In our work we propose to use task models of visual abstract tasks to generate scenarios that can be used either to validate a specific technique or to compare different techniques.

Patterns of abstract tasks have been identified in several fields such as multimedia systems [7] and Web applications [14, 3]. Byrne et al. [3] created a taxonomy of user tasks for the web, based on the analyses of most frequent tasks performed by users while using web applications. Those studies describe patterns for user tasks in terms of a taxonomy that can be employed to provide a comprehensive vocabulary for user activity in such application domains. The set of tasks presented by Garzotto et al. [7] has been employed to structure usability evaluations of multimedia applications. These tasks are described informally and in several aspects they can be considered more as guidelines or criteria for usability evaluation than actually user tasks.

None of these previous works present task models for interactive information visualization systems nor describe how patterns could be used during the development process or for evaluation purposes of an application.

The case study showed a simple application of our approach for building scenarios to compare the behavior of two different hierarchical visualization techniques. The advantage of using scenarios is evident when we notice that scenarios produced manually (see [11]) imply dealing with problems introduced by question and task specification formats. Scenarios produced informally are subject to adaptations that may introduce bias in the evaluation. Empirical testing and heuristics evaluation might not address all the situations that can be generated by the iterative and interactive nature of such techniques. The automated generation of scenarios guarantees that all the possibilities derived from the task model are present to be considered when transforming it in user concrete user tasks.

As discussed by Cockton et al. [4], improved evaluation occurs when evaluators are required to report and rationalize their use of heuristics, and of discovery method and confirmation/elimination knowledge resources. Task models not only can guide the rational design of applications but also can be used in later phases of development process to support usability evaluations, as proposed in this paper. Precise scenarios extracted from tasks models help evaluators to make a judgment on the effective use of a pool of well known visualization techniques.

Further work will be directed to model the complete set of abstract visual tasks, and applying them to case studies with different techniques. We also aim at investigating how to incorporate ergonomic criteria specialized for information visualization [6] in the scenarios generation procedure.

**ACKNOWLEDGMENTS**

This work has been supported by CAPES/COFECUB project SPIDER (grant nr. 399/02) and CNPq (Brazilian national science funding agency)

**REFERENCES**

1. Byrne, M.; John, B. E.; Wehrle, N. S.; Crow, D. C.
The Tangled Web We Wove: A Tasknomy of WWW


