35.1 Introduction

Computer users with motor impairments of upper limbs often face difficulties in accessing interactive applications and services using standard input devices. Several technological approaches attempt to deal with the issues raised by motor impaired users' interaction with graphical environments, each addressing a different user category according to their individual characteristics and abilities. One of these methods is scanning, which is mainly targeted to users with poor muscle control, weakness, fatigue, or severe mobility problems in general.

This chapter discusses the automatic hierarchical scanning technique for Windows applications, a method that enables motor-impaired users to work with any application running in Microsoft Windows, by using only a binary switch as an input device, thus overcoming problems related to the inability to use traditional input devices, such as the keyboard or the mouse. The sections that follow describe the scanning technique in detail, present a review of scanning approaches in research and industry, and introduce the automatic hierarchical scanning technique through the case study of a scanning tool named FastScanner.

35.2 The Scanning Technique

Scanning is an interaction method addressing the needs of users with hand motor impairments. The main concept behind this technique is to eliminate the need for interacting with a computer application using traditional input devices, such as with a mouse and keyboard. Therefore, all the interactive objects comprising a graphical user interface are being sequentially focused and highlighted (e.g., by a colored marker), while users can select to interact with the object currently having the focus by activating a switch. To eliminate the need for using a keyboard to type in text, an onscreen keyboard is usually provided.

During scanning, the focus marker scans the interface and highlights interactive objects sequentially, in a predefined order (e.g., from top to bottom and from left to right). Furthermore, scanning can be either automatic or manual. In the first case, the marker automatically moves from one interface element to the next after a predefined time interval of user inactivity (i.e., not pressing the activation switch), while the time interval can usually be customized according to user needs. In manual scanning, the user moves the focus marker to the next interface element whenever desired with the use of a switch.

Activation switches can vary from hand, finger, foot, tongue, or head switches to breath-controlled switches or eye-tracking switches (ABLEDATA, 2008; Enabling Devices, 2008). Figure 35.1 presents some indicative switches of the aforementioned types. Furthermore, any keyboard key (e.g., the space key) or mouse click can be used as a switch.

35.3 Review of Scanning Approaches

Scanning approaches can be classified in two main categories:

- Approaches targeted at embedding scanning in the development of software applications
- Approaches targeted at providing scanning techniques compatible for use with existing software applications

The first category follows a proactive philosophy (Stephanidis and Emiliani, 1999) targeted to application developers to create
accessible products and services without the need for a posteriori adaptation. This category can be further classified to:

1. Augmented libraries, enhanced with interaction objects inherently supporting scanning (Savidis et al., 1997). An example of an application based on this approach is discussed in Chapter 44, "Developing Inclusive e-Training," of this handbook. The main drawback of this approach is that application and services developed with such libraries face the problem of becoming obsolete once the next generation of the operating system they have been developed for is introduced.

2. Policy initiatives by mainstream actors, such as Microsoft Accessibility Developer Center (Microsoft Corporation, 2008a) and Sun Microsystems Accessibility Program (Sun Microsystems, 2008a), which facilitate software developers in creating more accessible products.

On the other hand, approaches addressing directly the needs of motor-impaired users themselves can be categorized according to the following criteria:

1. Accessibility features of operating systems
2. Applications with embedded scanning
3. Scanning tools, aiming to offer more generic solutions to users with hand motor impairments

Accessibility features of operating systems aim to make computer use easier for users with disabilities. Since most of them are not directly related to scanning, they are only shortly presented in this chapter for completeness purposes. The features that address the interaction requirements of motor-impaired users include (Linux Online, 2008; Microsoft Corporation, 2008b; Sun Microsystems, 2008b) but are not limited to: key combinations (e.g., Ctrl+Shift) that can be provided by pressing one key at a time rather than simultaneously, mouse cursor move and mouse button operations that can be carried out with the use of the keyboard, brief or repeated keystrokes that can be ignored and the repeat rate adjusted, onscreen keyboard with scanning facilities, options to adjust how fast a mouse button should be clicked for a selection to be made, and more.

Applications with embedded scanning are developed so as to support scanning in the first place and are accessible to people with motor impairments. However, they only partially address the interaction requirements of users. For example, a user needs more than one application to carry out a variety of everyday tasks (e.g., web browser, e-mail client, entertainment software, educational software, document authoring software, etc.). Therefore, users with motor impairments should employ various applications with embedded scanning techniques, possibly facing interoperability issues, and should often update to the latest version of each such application. Furthermore, a major drawback of these approaches is their increased cost of development and maintenance.

Finally, scanning tools enable users to operate the graphical environment of the operating system, eliminating the need for using various specialized applications for carrying out everyday tasks. Keyboard and mouse emulation programs with embedded scanning are popular among scanning tools, since they intend to ensure user interaction without the use of the traditional keyboard and mouse. Several research and industrial efforts have been targeted toward keyboard emulation, suggesting a variety of approaches, such as alphanumeric keyboards (Gnanayutham et al., 2004), chorded keyboards (Lin et al., 2006), keyboards enhanced with mouse emulation options (EnableMart, 2008) or with words and phrases abbreviation (Words+, 2008) and word prediction (WIViK, 2008) features. On the other hand, mouse emulation with scanning support software allows users to control the mouse pointer using one or two switches (Applied Human Factors, 2008; Gus Communications, 2008). Chapter 29 of this handbook discusses keyboard and mouse emulation in detail. Finally, tools that provide scanning of interface elements based on their screen location (Cooper and Associates, 2008) or their place in the hierarchy of objects comprising the active window (Ntoa et al., 2004), provide switch access to motor impaired users.
35.4 Automatic Hierarchical Scanning

The hierarchical scanning method provides access to all the interactive interface elements of the currently active window, based on their place in the hierarchy, by dynamically retrieving the window hierarchical structure. The main advantage of this technique in comparison to other scanning approaches is that it ensures rapid interaction, avoiding time-consuming sequential access to all interactive interface elements.

Issues related to hierarchical scanning are illustrated in this chapter through the case study of the FastScanner tool (Ntoa et al., 2004), which provides access to Microsoft Windows applications without recourse to any subsequent modification, through the use of binary switches as an alternative to traditional input devices. In the following subsections, the FastScanner architecture is presented, the hierarchy retrieval and filtering process is analyzed, the classification of interactive objects into categories of scanning objects is explained, and an example of scanning dialogue is provided. Finally, issues related to the tool's graphical user interfaces as well as evaluation issues are discussed.

35.4.1 FastScanner Architecture

FastScanner retrieves the initial hierarchy of the objects comprising the active application window through the Microsoft Active Accessibility platform, as presented in Figure 35.2. However, since the initial hierarchy retrieved is radically different from the actual hierarchy of interface elements that are visible to the user, a filtering stage is required before establishing the final objects’ hierarchy. Once the final hierarchy is determined, the scanning dialogue is initiated, having the window itself as the currently focused and highlighted object.

As shown in Figure 35.2, in the context of the FastScanner tool, all the interactive objects that may compose an application have been classified into categories, and their behavior during a scanning dialogue has been modeled using State Transition Networks (Parnas, 1969; Wasserman, 1985). Furthermore, user input is interpreted by the scanning dialogue management unit according to the type of the object currently in focus, and the corresponding output is produced based on the state transition networks. Finally, when user interaction causes a transformation in the current application hierarchy, the application itself notifies the Active Accessibility platform, through the Windows kernel. The Microsoft Active Accessibility platform in turn notifies the FastScanner application, which reconstructs appropriately the final objects’ hierarchy.

35.4.2 Hierarchy Retrieval and Filtering

The objects’ tree-structured hierarchy in FastScanner is retrieved through Microsoft Active Accessibility, having as root the active application window itself. During this process, once an object is added to the hierarchy, all of its children are examined and if necessary added to the hierarchy. To add an element to the hierarchy, the following properties are retrieved and stored:

- **Name**: It should be noted that an object’s name cannot effectively identify it, since there are several nameless window objects. For example:
  - The title bar of a window is a nameless object.
  - The minimize button of the title bar has the name Minimize.
- **Description**: It is a short text providing a brief description of the objects’ visual appearance. For example:
  - The description text for a window title bar is as follows: “Displays the name of the window and contains controls to manipulate it.”

It should be mentioned that this attribute is empty for many objects as well.

- **Role**: It is a property of all windows objects, explaining their interactive role. For example:
  - The role property for a window title bar is “title bar.”
  - The role property for a button is “push button.”
- **Value**: This property is used only in some objects, when there is a need for value specification. For example:
  - The value attribute of a scroll bar represents the bar place as a percentage, from 0 (top) to 100 (bottom).
- **Action**: This attribute is empty for noninteractive objects, while for interactive objects it describes the action that a user can perform on them. For example:
  - The action attribute for a button is “Press.”
  - The action attribute for a menu is “Open.”
- **State**: There are several values for this attribute, describing objects’ state. Examples of state values are as follows:
  - “Normal,” when no special state should be indicated.
  - “Unavailable,” for objects that are not interactive.
  - “Invisible,” for objects that are not visible.
• “Selected,” for checkboxes that are selected.
• Coordinates: Refers to information regarding the screen location of an object by recording the top left corner coordinates as well as the object width and height.
• Number of children.
• Window handle: This attribute is provided by the operating system and can uniquely identify a window.

It should be noted that the objects comprising the final hierarchical structure are not only the interactive window objects, but container objects as well, whose role is to group the objects contained. Examples of such container objects are windows, group boxes, and frames. Although in graphical user interfaces (GUIs) no user interaction is supported for container objects, in scanning dialogue they are of great importance, as they accelerate interaction by allowing users to directly skip the scanning of large groups of objects.

Additionally, before actually adding an object to the hierarchy, and once its properties have been retrieved, they are examined to determine whether the object should be actually added to the hierarchy. In more detail, the object attributes that are relevant are:

• Role: There are several objects that can be directly excluded from the final objects’ hierarchy due to their role, which reveals that user interaction is not possible with these objects. Such object roles are: alert, animation, border, caret, character, chart, clock, cursor, diagram, dial, grip, equation, graphic, helpballoon, hotkeyfield, indicator, progressbar, rowheader, separator, slider, sound, statusbar, tooltip, and whitespace. Furthermore, static text is also excluded from the hierarchy, since no user interaction is involved.
• State: By assessing an object’s state it can be determined whether it is visible and available for interaction. If these two conditions are not satisfied, the object is not added to the final hierarchy.
• Coordinates: Several objects, although having a valid role and state, are actually not visible to the user, due to their placement. By inspecting their screen location, all these objects are eliminated from the hierarchy.

Finally, if during the aforementioned checks, it is determined that an object should not be included in the hierarchy, its parent’s “number of children” attribute is reduced by one. In the case where the parent node is left without any children at all and it is not an interactive object, then the parent node is also removed from the hierarchy and its parent is recursively updated. In the case where the parent node is not interactive and left with one child it remains in the hierarchy. However, in this case, an extra object attribute is added, indicating that the scanning dialogue should not affect this object, but directly its child.

As a result, after the tree enhancement process, where all the unnecessary nodes are eliminated, the final object hierarchy includes only the necessary interactive elements. To better illustrate the tree enhancement process, an example is presented.

**FIGURE 35.3** Example window for hierarchy retrieval.

The initial objects’ hierarchy for a window consisting of a text field and two buttons, as presented in Figure 35.3, consists of 142 objects. On the other hand, the final enhanced objects’ hierarchy consists of just 10 objects. The initial hierarchy retrieved and the enhanced one are represented in Figures 35.4 and 35.5, respectively.

### 35.4.3 Interactive Objects’ Categories

Interactive objects’ classification into categories according to their behavior in scanning dialogue leads to the design and development of state transition networks, modeling objects’ interactive behavior. This is a significant unit of the FastScanner architecture, since it allows the effective and efficient management of the scanning dialogue. The objects’ categories as defined for the FastScanner application are:

• Text input objects: They represent interface elements used to provide text to an application, such as text fields. To address the needs of motor-impaired users, a virtual keyboard supporting scanning has been designed and embedded in FastScanner.
• Simple objects: They represent interface elements that are directly associated with a single user action. When using the traditional input devices, interaction with these objects is achieved with a single mouse click. Well-known examples of such elements are buttons.
• Selection objects: These objects usually belong to a list or a tree structure, and users need to select the objects before interacting with them. When using the traditional input devices, selection of these objects is achieved with a single mouse click, while interaction with them is achieved with a double mouse click.
• Container objects: Their role is to group other interactive elements. When using the traditional input devices, these objects do not entail any user interaction at all.

Table 35.1 represents the most typical examples of objects belonging to each category.

Since each one of the previous object categories is characterized by a different interactive behavior, and to better address the needs of scanning users, a different scanning dialogue technique was designed and implemented for each object category. Nevertheless, all the interaction techniques are based on the
same interaction principles concerning dialogue states and user actions. In more detail, there are three types of dialogue states that are supported in FastScanner and that are explicitly indicated to the users by the color of the focus marker:

- **Entry state**: It is represented by the green color of the marker and indicates that the object is active and ready to be used.
- **Exit state**: It is represented by the red color of the marker and indicates that the dialogue is ready to move to the next interactive interface element.
- **Selection state**: It is used only in the case of selection objects, such as the items of a list; it is represented by the orange color of the marker and indicates that the enclosed object is ready to be selected.

On the other hand, user input is provided through the use of switches. FastScanner has been designed to support three types of switches, and therefore three types of input commands:

- **Select**: Initiation of the selection action indicates that the user wishes to proceed with the action that the dialogue state indicates. Therefore, if the scanning dialogue state is entry, the user will interact with the current object. Otherwise, if the scanning dialogue state is exit, the dialogue will move to the next interactive interface element.
- **Next**: Initiation of the next action indicates that the user wishes to change the dialogue state to the next available one.
- **Reverse**: Initiation of the reverse action indicates that the user wishes to reverse scanning order from top-down to bottom-up and vice versa.

Furthermore, the possibility for time scanning has been added to FastScanner to help users interact with a Microsoft Windows application more efficiently. When working in time-scanning mode, users have to initiate only the select actions. The next action is automatically initiated by the system when a specific time interval elapses without a user action.

Based on these scanning dialogue principles and the interactive properties of each object category, as well as model users' interaction with an application using FastScanner, a state transition network (STN) diagram has been designed and implemented for each object category. All the STN diagrams are represented in Figures 35.4 through 35.9. In general, for all object categories, the dialogue is initiated in exit state and the following general principles are followed:

- If the dialogue is in exit state and the user initiates the select action, then scanning focus moves to the next object in the scanning hierarchy.
### TABLE 35.1 Classification of Windows Object Types

<table>
<thead>
<tr>
<th>Text Input Objects</th>
<th>Simple Objects</th>
<th>Selection Objects</th>
<th>Container Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text field</td>
<td>Drop-down menu buttons</td>
<td>List items</td>
<td>Title bars</td>
</tr>
<tr>
<td>Text area</td>
<td>Menu buttons</td>
<td>Combination list items</td>
<td>Menu bars</td>
</tr>
<tr>
<td></td>
<td>Buttons</td>
<td></td>
<td>Scrollbars</td>
</tr>
<tr>
<td></td>
<td>Menu items</td>
<td></td>
<td>Group boxes</td>
</tr>
<tr>
<td></td>
<td>Option buttons</td>
<td></td>
<td>List boxes</td>
</tr>
<tr>
<td></td>
<td>Checkboxes</td>
<td></td>
<td>Drop-down list boxes</td>
</tr>
<tr>
<td></td>
<td>Spin buttons</td>
<td></td>
<td>Drop-down menus</td>
</tr>
<tr>
<td></td>
<td>Links</td>
<td></td>
<td>Page tab control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tables</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Windows</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dialogue boxes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frames</td>
</tr>
</tbody>
</table>
In container objects, if a select action is initiated when the dialogue state is entry, then the scanning focus moves to the first contained object.

Besides the previously mentioned STNs, some additional ones implementing scanning dialogue for more experienced users were designed as well, aiming to accelerate user interaction. The quick dialogue STNs are presented in Figures 35.10 through 35.13.

The main difference is that quick scanning dialogue does not include the exit state at all. In more detail, when scanning focuses on an object, the dialogue is initiated at entry state. If the user presses the switch corresponding to the select command, then the appropriate interaction according to the object type will be initiated. In the case of text entry objects, interaction entails displaying the virtual keyboard, while in the case of container objects interaction means moving the scanning focus to the first contained object. Since there is no exit state, when the next command is provided in entry state, the dialogue moves to the next object in the case of text entry, simple and container objects. In the case of selection objects, the dialogue moves to the next available state (i.e., the selection state).

In conclusion, the tool supports two types of user input, namely, time-based and manual, and two modes of function, namely, standard and quick scanning. The summary of these operation modes is provided in Table 35.2.

### 35.4.4 Scanning Dialogue Example

After having discussed the FastScanner object types and how user interaction has been modeled according to the object type, and to better illustrate the functionality of FastScanner, a step-by-step example is presented in this section. The example refers to the scenario of opening the folder “My Documents” using the quick dialogue mode.

---

**FIGURE 35.7** Dialogue STN for simple objects.

**FIGURE 35.8** Dialogue STN for selection objects.
**FIGURE 35.9** Dialogue STN for container objects.

**FIGURE 35.10** Quick dialogue STN for text entry objects.

**FIGURE 35.11** Quick dialogue STN for simple objects.

**FIGURE 35.12** Quick dialogue STN for selection objects.
1. Scanning dialogue initially focuses on the window itself (i.e., the Microsoft Windows Explorer window).
2. Following a select command, scanning focuses on the first contained object.

3. The next action moves the focus marker to the next element, for instance, the title bar, in the entry state.
4. Since there is no need to interact with one of the objects contained in the title bar, the next action is initiated and the dialogue focuses on the next object (the toolbars container), in the *entry* state.

5. Since there is no need to interact with one of the objects contained in the toolbars container, the next action is initiated and the scanning dialogue focuses on the next object (the main window area), in the *entry* state.
6. Since the target object is included in this container, the *select* action is provided and the dialogue moves to the first contained element.

7. Once again, since there is no need to interact with one of the objects contained in the highlighted area, the *next* action is initiated and scanning focuses on the next object, which is the target object, i.e., a link to the folder "My Documents."

8. Following a *select* command, scanning focuses on the next object, which is the target object (i.e., a link to the folder named "My Documents").
9. By triggering the select command, the currently active object will be activated and the folder named "My Documents" will be opened.

35.4.5 Graphical User Interfaces

The graphical user interfaces of the tool aim to allow users to perform actions that are usually carried out with the use of a pointing device, such as moving or resizing a window, as well as to personalize their interaction with the tool.

In more detail, the tool should provide users with a way to select which one of the currently opened windows is the one they wish to interact with. Therefore, when the user initiates the FastScanner tool, a window appears, presenting all the application windows that are currently open, and asks the user to select the desired application. This dialogue is depicted in Figure 35.14a. Interaction with the initial dialogue of the tool takes place through scanning, and with the use of the binary switches, as described earlier. Once the user selects the Scan button, the requested application comes to the foreground and the scanning dialogue is transferred to it.

Furthermore, one additional dialogue box is used for window moving and resizing actions, as well as for activating the scanning focus marker and scanning input settings dialogues. The main settings dialogue box is presented in Figure 35.14b and can be triggered by pressing the select and next switch at the same time in manual scanning mode, or by activating the next switch in time-scanning mode. In more detail, the user can move the

![Figure 35.14](https://example.com/figure35.14.png)

**FIGURE 35.14** GUI elements of the tool: (1) initial window, (2) settings dialogue, (3) border settings (color and width), and (4) input device settings.
window of the application by selecting one of the eight buttons of the first two rows, or resize the window by selecting one of the four buttons of the third row of the main settings dialogue. The button with the palette initiates the border settings window, which is displayed in Figure 35.14d and allows the user to set the color for each one of the dialogue states (entry, exit, and selection), as well as the width of the focus marker. The button with the input device in the initial settings window initiates the input device settings window, which is displayed in Figure 35.14c and is used for enabling or disabling the time-scanning mode and specifying the time interval for the automatic triggering of the "next" action. The last two buttons of the initial settings window can be used for pausing and restarting FastScanner and for terminating the operation of the tool.

Finally, one additional window that was designed and is embedded in the FastScanner tool is that of the virtual keyboard, which is presented in Figure 35.15. The virtual keyboard was created using the QWERTY layout, by omitting the numeric pad keys and enhancing it with an additional window functionalities toolbar. The functionalities toolbar provides controls for the management of the keyboard window itself. In more detail, the first four buttons move the keyboard left, right, up, and down, the next four buttons resize the window and the last button closes the virtual keyboard window. Finally, the button named English above the Insert, Home, and Page Up buttons is used to change the keyboard language from English to Greek and vice versa.

User tests carried out for the keyboard indicated that users were rather slow in typing text. Therefore, buttons were grouped into clusters, as shown in Figure 35.16. User tests of the enhanced virtual keyboard with groups provided better results, which were further optimized for experienced users who would work effectively with the reverse switch as well.

35.4.6 Events Handling

As windows change their structure and their properties according to user actions, it was important to monitor user actions and interpret their results on the currently active window to update the scanning hierarchy whenever necessary. More specifically, whenever a user acts on a windows object, the operating system sends a message to all the applications that are hooked to the events of a specific window. This message consists of certain parameters, defining the event, the object that generated it (by providing its window handle), as well as the object type.

FastScanner hooks the event messages posted for the currently scanned window and acts accordingly when necessary. For example, if the message indicates that the window has been moved, all the object properties of the scanning hierarchy, referring to the objects' location on the screen, are updated to the latest information. As a result, when necessary, the objects' hierarchy is being traversed and the necessary objects are added, updated, or deleted.

It should be noted, however, that the operating system posts several messages for one user action performed on a windows object, and therefore these messages have to be filtered and interpreted according to the current context. For example, when a window comes to the foreground, four messages are posted: EVENT_SYSTEM_FOREGROUND, EVENT_OBJECT_LOCATIONCHANGE, EVENT_OBJECT_REORDER, and EVENT_OBJECT_FOCUS. However, the hierarchy should be constructed only once to achieve a fast system response time. Events handling in FastScanner is a complex process and required prioritizing the importance of the posted messages based on their significance, their uniqueness, and their consistency to avoid unnecessary hierarchy reconstruction.

35.4.7 Evaluation

To assess the usability of the FastScanner tool, a user-based laboratory usability evaluation has been performed, involving ten participants, five with motor disabilities of upper limbs and five able-bodied users simulating temporary inability to use their hands (Ntou et al., 2004). The evaluation was based on an appropriate scenario of common tasks performed by a typical computer user, such as writing documents, reading and sending an e-mail, and navigating in the web. An analysis of the performance results indicated that participants were in general effective in carrying out the tasks. In particular, they achieved the majority of the requested task goals, and the number of errors was small in average. Additionally, the majority of errors were corrected, while rarely the users asked the observer for help. Furthermore, participants' interaction with the FastScanner application was considered rather efficient, since they did not need a very long
time to carry out the requested tasks, and they rarely needed to consult the instruction booklet. Finally, the analysis of the users' responses to the satisfaction questionnaires suggested that they were overall satisfied by the tool. Participants were on average less satisfied during the execution of the first scenario, entailing text typing. A detailed inspection of their answers to the questionnaires revealed that they met difficulties in using the virtual keyboard to enter text. However, it was noticed that able-bodied users were more dissatisfied than disabled users, probably due to the fact that they are acquainted with more rapid interaction for entering text.

35.5 Summary and Conclusions

This chapter discussed current approaches to scanning and has thoroughly presented an approach to automatic hierarchical scanning in a windows environment through the case study of a tool developed according to this method, the FastScanner tool. FastScanner allows user interaction through the use of binary switches, and by providing the appropriate graphical user interfaces users can work efficiently with any application they wish.

To achieve fast user interaction, FastScanner adopts the hierarchical scanning technique and provides access to the objects of the selected window according to their place in the window hierarchy and not based on their screen location as in row-column scanning. User interaction is also accelerated by the use of container objects, allowing users to skip scanning large groups of objects. Furthermore, the tool supports two types of user input, namely time-based and manual, and two modes of function, namely standard and quick scanning. In time-based input, users are able to interact with the use of just one switch. Standard scanning mode addresses first-time users of the tool, while quick scanning mode serves the needs of expert users.

Evaluation of the tool indicated that participants were in general effective and efficient in carrying out the tasks. In particular, they achieved the majority of the requested task goals, the number of errors was small in average, and they completed the tasks in reasonable time without the need for any help. Finally, the analysis of the users' responses to the satisfaction questionnaires suggested that they were overall satisfied by the tool, although they were first-time users.

References


