Univ Access Inf Soc (2010) 9:77–86
DOI 10.1007/s10209-009-0155-2

LONG PAPER

# GOMS analysis as a tool to investigate the usability of web units for disabled users

**Martin Schrepp**

Published online: 1 August 2009


**Abstract** Guideline compliance is a necessary but not sufficient condition to guarantee the usability of web units by disabled users, since efficiency-related issues can be as exclusive for disabled users as violations to basic guidelines. This paper shows that Goals, Operators, Methods and Selection rules (GOMS) analysis, which is an established method in user interface design, can be adapted to evaluate the efficiency of interface designs for disabled users. As examples, several GOMS models for the interaction behavior of disabled users with web units are described, showing how such models can be used to answer concrete accessibility-related questions. Advantages and limitations of GOMS analysis are also discussed.

**Keywords** GOMS analysis · Efficiency · Usability · Accessibility

## 1 Introduction

The basic idea of *Design for All* (also known as *Universal Design* or *Barrier free Design*) [1, 2] is to create user interface designs which as many people as possible can use either directly or with the help of assistive technology.

Web units[1] designed under this concept must guarantee a certain degree of usability for all user groups, including users with disabilities. As a consequence, the issue arises of evaluating how well a web unit fits to the *Design for All* concept, and especially how usable and accessible pages in the web unit are for different groups of disabled users.

M. Schrepp (✉)
SAP AG, Walldorf, Germany
e-mail: martin.schrepp@sap.com

The most common approach is to test automatically or manually if the web unit is conform to published accessibility guidelines, for example Sect. 508 [4] or WCAG 1.0 [5]. A number of automatic check tools for web pages can be used (for example [6–8]). Automatic checking against guidelines is a very efficient method to detect accessibility problems in web units, but it is restricted to checkpoints which do not require human judgement. It is, for example, possible to check automatically if a graphic on the page has an alternative text. But it requires human judgement to assess if this alternative text adequately describes the content of the graphic.

Manual tests against published guidelines avoid these issues, but the costs for a manual test increase of course with the number of pages that have to be checked. Thus, for web units containing a higher number of pages manual tests must concentrate on a small sample of pages. Conclusions concerning the accessibility state of the other pages in the web unit must be drawn from the results of this sample.

Additionally, focusing on a check against guidelines reflects a somewhat restricted concept of accessibility. Some authors call this the technical accessibility definition [9]. This is opposed to a more usability[2] oriented definition, which states that disabled persons should be able to use

[1] A web unit is defined by the WCAG 2.0 Draft [3] as a collection of information, consisting of one or more resources intended to be rendered together, and identified by a single uniform resource identifier. We use the term web unit throughout the paper to refer both to classical Web sites with more or less static pages and to dynamical web applications.

[2] We use the term usability in the sense of ISO 9241 [13], where it is defined as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

a web unit with effectiveness, efficiency and satisfaction. Such a usability oriented approach allows transferring established usability methods to the area of accessibility. Examples are user modeling techniques [10], usability review methods such as cognitive walkthroughs [11], or cognitive task analysis [12]. In general, such approaches take also the cognitive aspects of accessibility into account.

Another method to check a web unit concerning accessibility is to test it with disabled users. User tests allow detecting problems which may not be visible through a pure guideline check. A study by the British Disability Rights Commission [14] found, for example, that 45% of the problems encountered by disabled users when attempting to navigate Web sites cannot be attributed to explicit violations of the WCAG 1.0 guidelines.

A drawback of usability testing is that it is in most cases very expensive. First, it is difficult to find the right test persons. Second, tests must ideally be performed with users showing different types of disabilities, since depending on the disability different problems may show up. Thus, in practice usability tests with disabled users are difficult to organize given today's short development cycles for software.

Another problem with usability tests including disabled users is that they can only be performed very late in the production process of a web unit. The reason for this problem is that such tests require an already running system or a very carefully developed high fidelity prototype. This is a crucial difference to usability testing with non-disabled users, for which often a paper prototype is sufficient to get valuable insights into interaction design problems. But such low fidelity prototypes can obviously not be used for people who rely on assistive technology.

Usability tests with disabled users can be very effective to develop general insights into problems of specific user groups, but not for quality enforcement concerning all pages[3] in large commercial web units and not as an effective feedback mechanism for designers in the early design phase.

An often mentioned problem of guideline-based evaluations of web units is that this method tends to ignore efficiency-related issues in interface design [15–20]. Guidelines currently focus on the existence of concrete barriers in web units, i.e., on problems which hinder disabled users to perform the relevant actions in the web unit. But it is important to understand that a web unit can be fully compliant with guidelines, but practically unusable for specific groups of disabled users[4] [17, 18].

For example, web units which are used for professional work require that users can perform tasks in a certain time limit. If a user with a disability needs ten times longer to finish a standard task (which he or she may need to perform very often during a working day) than a non-disabled user, then the user will not be able to work efficiently with the web unit, even if the web unit is fully compliant to accessibility guidelines.

Another example of a serious efficiency-related accessibility problem can be found in many large Web sites. Such Web sites contain often a huge entry page, which is packed with a huge number of links (often more than 100) [18]. If such entry pages offer no page internal shortcuts, or a page internal search function, or a good structure of the content over appropriate headings, it is for blind users (also for sighted users navigating by keyboard) very time consuming to reach the information they search for. If the time required to reach relevant links from the top of the page exceeds a certain time limit, then the relevant information is practically unusable and inaccessible for blind users, even if they are in principle able to reach it.

Such efficiency-related issues can thus be as crucial for the ability of disabled users to work with web units as concrete violations of high priority guidelines. Thus, compliance with guidelines is a necessary but not sufficient condition to ensure that a web unit is practical accessible and usable by disabled people [14].

Designers of web units should thus spend more efforts to analyze potential efficiency-related issues and to design products in such a way that allows disabled users to perform all relevant tasks with sufficient efficiency [15, 18, 19].

A well-established tool to investigate efficiency-related design problems is Goals, Operators, Methods and Selection rules (GOMS) analysis. With the help of GOMS it is possible to model the interaction behavior of users with software products for goal-directed routine tasks. GOMS analysis is thus a tool to analyze problems with user performance early in the design phase with acceptable costs. Currently, most practical applications of GOMS analysis describe the behavior of non-disabled expert users of a system. This paper will show that it is possible to adapt existing models to describe also the interaction behavior of users with disabilities and to formulate models which explicitly describe the interaction of special groups of disabled users, for example blind users [19, 20].

[3] If the navigation and appearance of all pages in the web unit is based on common mechanisms (for example, if the pages are generated automatically from a content management system), then a usability test of these common mechanisms by disabled users can be of course an efficient way to improve the accessibility of the whole Web site.

[4] Some authors, for example [17], point to the fact that web units not compliant to accessibility guidelines can be in fact quite usable by disabled users in practice. Thus, the relation between compliance to accessibility guidelines and the usability for disabled people seems to be more complex than expected.

This paper will also discuss the additional benefits of using GOMS analysis, or more general engineering models of human performance,[5] to investigate the adequacy of interface designs for disabled users. In addition, it will discuss the limitations of such approaches.

## 2 GOMS models

GOMS analysis was developed as a quantitative method to access and compare the efficiency of different user interface design alternatives [21]. A GOMS analysis allows to predict how long an experienced user needs to perform a given goal-directed task in a given interface design.

The acronym GOMS stands for Goals, Operators, Methods and Selection rules, which are the basic building blocks of such a model:

- A goal describes what the user wants to accomplish.
- Operators are basic physical or cognitive processes, which must be performed during the process to reach the goal.
- A method is a sequence of operators. Methods represent basic learned operator sequences users perform to reach a goal or sub-goal.
- If at a point in time more than one method can be performed to come closer to the goal, then selection rules decide which of these possible methods is chosen.

Operators represent the basic physical or cognitive processes that a user must perform in a given task. They are the basic building blocks in a GOMS analysis. Examples of typical physical operators used in GOMS analysis are:

- tap a key on the keyboard,
- move the hand from the mouse to the keyboard or vice versa,
- position the mouse cursor to a target on the screen.

Examples of typical cognitive processes used in GOMS analysis are:

- recall a piece of information from memory,
- decide between two alternatives,
- prepare mentally for the next step in a sequence.

Different users will need different times for these elementary operators. However, for a comparative analysis of screen designs it is sufficient to use a set of typical or average operator times. These average times are typically determined in laboratory experiments [22–25]. They represent the average performance of trained non-disabled users of a system.

Examples of typical times used in GOMS analysis are:

- tap a key on the keyboard while typing a string: 0.2 s,
- position the mouse cursor to a target on the screen: 1.1 s,
- mental preparation for the next step in a sequence: 1.35 s.

There is some freedom for the researcher to decide on the granularity of a GOMS analysis. A step which is represented by a method (i.e., a sequence of operators) in one model can be represented as a single operator in another model.

There are several variants of GOMS analysis available. These variants differ as for the complexity of the analysis and the accuracy of the predicted task completion times. A detailed comparison of these variants can be found in [22, 23].

The keystroke level model (KLMGOMS) is the simplest form of GOMS. This variant does not use the higher level cognitive modeling capabilities of GOMS. It uses mainly the physical and cognitive operators required to finish a task to predict the task completion time. Thus, goals, methods and selection rules are not used in KLMGOMS, which simplifies the modeling process significantly. The estimated task completion time depends thus only on the physical and cognitive operators required to finish a task.

When analyzing a task through KLMGOMS it is generally easy to determine the required physical operators and their order. The correct placement of cognitive operators in the sequence is much more difficult. This is typically done by splitting the task into several cognitive units. Then a cognitive operator (which accounts for example for mental preparation or decision making) is placed in front of each cognitive unit. This approach requires a deeper understanding of human information processing in the given task and is thus the most critical part of a KLMGOMS analysis.

One of the serious limitations of KLMGOMS is that this model does not allow describing multi-tasking behavior, i.e., the fact that human information processing can run several processes in parallel. The Critical-Path Method GOMS (CPMGOMS) [22, 24] was designed to overcome this limitation. Like in a KLMGOMS analysis, the task is represented as a linear sequence of operators. But then this sequence is split into subsequences of cognitive units, subsequences running in parallel are identified. Thus, the task completion time does not only result from the single times of the involved physical and cognitive operators, but also from an in-depth analysis of the cognitive processing of the task. The predicted task completion times of a correct CPMGOMS analysis should therefore be more accurate than the times predicted from a comparable

[5] These are formal models of human performance in human–computer interaction tasks, which allow quantitative predictions concerning variables such as, for example, task completion time, learning time, or error rates.

KLMGOMS analysis. But this comes at the cost of a much higher difficulty of the analysis.

Another GOMS variant which must be mentioned is the Natural GOMS Language (NGOMSL) [25]. NGOMSL is a syntax for GOMS models, which formalizes the way GOMS analysis is performed. NGOMSL make it possible to estimate not only possible task completion times, but also learning times for given tasks.

This paper focuses on KLMGOMS, since the accuracy of the predictions of this method seems to be sufficient for the set purpose, and modeling concrete tasks is relatively simple.

Several studies, see for example [27], have shown that carefully constructed GOMS models can be used for an accurate prediction of the average task completion times of expert users of a system. Thus, in principle a GOMS analysis can be used to judge if users can reach predefined performance goals for standard tasks.

## 3 GOMS analysis and efficiency-related accessibility problems

In principle GOMS can be used to answer two different questions concerning the usability of a user interface by disabled users.

First, it can be investigated if the overall design of the user interface itself forces an unacceptable disadvantage for disabled users. For example, the dominant design principle for large Web sites is to organize the content of the site in a few overview pages, which contain a huge number of links. This design principle is based on the paradigm that it is more important to reduce the number of clicks to reach an end node in a hypertext structure than to control the complexity of the single pages (this is often summarized as *breadth is better than depth*).

Web sites constructed according to this design principle are heavily optimized for users who navigate with a pointing device. If a Web site contains a huge number of pages, then this design principle puts an unacceptable burden to sighted users who navigate the site by keyboard [18] or to blind users who access the content of the page using a screen reader [15]. In order to open the desired page, keyboard users have often to press the TAB key more than 100 times. Models of keyboard navigation in the web [18, 26] can help to quantify this problem and to evaluate the efficiency of additional mechanisms for keyboard support. Such mechanisms can then be implemented in addition to the standard TAB-chain.

One method to evaluate if the design of a web unit causes a disadvantage for certain user groups is a comparative analysis. For example, a potential objective could be finding out if the design of a web unit causes an unacceptable disadvantage for users who cannot use a mouse and are thus forced to navigate using the keyboard. A simple approach to answer this question is to construct models for keyboard and for mouse navigation in this web unit. With the help of these two models the task completion times for standard tasks in the web unit can be estimated for users navigating with the mouse and users navigating with the keyboard. A comparison of the estimated task completion times for both user groups shows if the disadvantage for keyboard users is critical or not.

Second, GOMS models can be used to evaluate if disabled users are in principle able to reach predefined performance goals. For example, a company may consider introducing new software, but some of the employees cannot use a pointing device. A GOMS model describing the interaction of users who navigate the web unit by keyboard can then be used to evaluate how long such users will need for their standard data entry tasks with the new software. It can then be decided if the new software allows such users to work productively.

The available operator times for GOMS analysis describe average times for highly trained non-disabled computer users. For the second use case mentioned above it is of course necessary to adapt the published operator times for GOMS analysis to times realistic for the intended target group, i.e., average times for trained computer users who have a certain disability. For example, the average time to tap a key on the keyboard used in most GOMS analyses ranges between 0.2 and 0.28 s. But [28], for example, reports average times of 0.6 s to tap a key on the keyboard for manually disabled users.

## 4 Engineering models describing the interaction of disabled users

This section presents three models which are especially formulated to describe the interaction of disabled users with a web unit. The first model focuses on users who cannot use a pointing device and must therefore navigate a web page using the keyboard. The second model describes navigation and data entry tasks in web applications. The third model describes the interaction of blind users in web units.

### 4.1 A model for keyboard navigation in Web sites

The structure of many large Web sites is highly optimized for sighted users who navigate with a pointing device [18]. Even if such sites can be handled by keyboard, it is often very time consuming and inefficient to navigate with this method. A model for keyboard navigation was constructed to quantify the disadvantage for users navigating the Web site using the keyboard.

The standard method to access links over the keyboard is provided by including these elements into the TAB-chain. Thus, all links in the page are organized into a virtual chain. The user can navigate to the next element of this virtual chain by pressing the TAB key. In addition, it is possible to navigate to the previous element on the chain by pressing the key combination SHIFT + TAB.

This method to make Web sites accessible through the keyboard is problematic if a Web site contains a huge number of links, because the user has to press the TAB key very often to reach the required element [18, 29].

Assuming that the goal of a user is to follow a link on a given page, if the user navigates using the keyboard and the target link is the $n$th link in the TAB-chain, the task can be split into the following sub-tasks:

- locate the target link on the page,
- press $n$-times TAB to place the cursor focus on the link,
- follow the link by pressing ENTER.

A typical problem when users navigate through long TAB sequences is that they sometimes lose the information about the current cursor position. In such situation, the user has to scan the page to detect the current focus position again.

It is worth noting that GOMS models do not allow to describe such more or less random errors in user interaction. Thus, the model described in the following is not a pure GOMS model, but enhances GOMS with a mechanism to describe random errors by the user. With the exception of this modification, the other parts of the model are compatible with a GOMS model.

The task analysis described above was used in [29] to formulate a model for keyboard navigation in web pages. The central assumption of this model is that the total time $t$ to detect and follow a target link on a page is given by:

$$t = t_1 + n \times t_2 + n \times p \times t_3 + t_4. \quad (1)$$

Here $n$ is the position of the target link in the TAB-chain. The free parameters have the following meaning:

- $t_1$ is the time required to locate the target link on the page,
- $t_2$ is the time required to press the TAB key,
- $p$ is the probability to lose the cursor focus during navigation,
- $t_3$ is the time required to locate the lost cursor focus again,
- $t_4$ is the time required to decide if the focussed link is the correct target link and to press ENTER.

The times for the operators in the model were estimated in a laboratory experiment. Ten experienced computer users (age ranged from 25 to 43 years) participated in this experiment. The task of the participants was to navigate to a link in a web page using only the keyboard. Each participant had to solve this task for ten web pages. These pages were selected per participant randomly out of a larger sample of common German web pages (from companies, public organisations, web shops, etc.).

The number of links on these pages varied between 38 and 200. The number of TAB presses to reach the target link from the top of the page varied between 14 and 119. Before the participants started the experiment they could practice keyboard navigation.

All interactions of the participants and especially the times between keyboard interactions were recorded by a tool installed locally on the test computer. Given these data, it was possible to provide valid estimations for the free parameters of the model (for details of the statistical analysis, see [29]).

The following estimates for the parameters were calculated from the data:

- $t_1$: 9.6 s,
- $t_2$: 0.27 s,
- $t_3$: 2.61 s,
- $t_4$: 1.13 s,
- $p$: 3.18%.

A detailed description of the experiment and the estimation of the parameter values can be found in [29].

Focus losses occur relatively seldom (in only 3.18% of the cases). But they cause a considerable delay, since it takes comparatively long (2.61 s) to detect the lost focus again. The most time consuming part of the navigation task is the initial location of the target.

The calculated times for the parameters of the model are valid for non-disabled highly trained computer users. Disabled users who are unable to use a mouse will be much slower in hitting a key. But the model can easily be adapted for such users. This can be achieved by replacing the operator times $t_2$ and $t_4$ with times realistic for the intended user group.

Which conclusions can be drawn from this model? First, one can calculate, for a web page which contains no page internal links or shortcuts (for example access keys), the average time a sighted keyboard user needs to detect and reach the $n$th link in the TAB-chain as

$$t(n) = 9.6 + n \times 0.27 + n \times 0.0318 \times 2.61 + 1.13. \quad (2)$$

This provides quantitative information concerning the possible number of links which can be placed on a web page without providing special access mechanisms for keyboard users.

It is also possible to compare the time required to navigate to the $n$th link by mouse and keyboard. In [26], a simple GOMS model for mouse navigation was described. The time to navigate to a link on a page (assuming that the

link is visible, i.e., the user has not to scroll down to reach the link) is estimated as

$$t = m_1 + m_2 + m_3. \quad (3)$$

In this equation

- $m_1$ is the time to locate the target link in the page,
- $m_2$ is the time to position the mouse cursor to the target link,
- $m_3$ is the time to decide if the target link is the correct link and to click on this link.

An estimation in the above-mentioned study showed that the parameters of this GOMS model can be set to $m_1 = 9.6$, $m_2 = 1.1$, and $m_3 = 1.13$.

For example, in a web page with 100 links which has not implemented any shortcuts or access keys, keyboard navigation must be performed solely over the TAB-chain. Assuming in addition that users of this web page select all links with the same probability, according to the proposed GOMS model for mouse navigation the average time to detect and follow a link on this page using the mouse is 11.83 s, whereas using the keyboard is 28.56 s. If the design target is set that the performance of keyboard navigation should not exceed two times the performance of mouse navigation, it can be concluded that additional keyboard shortcuts must be implemented for this page.

It is important to note that the model described above cannot be used to make accurate predictions of the time required to detect and follow a link by keyboard navigation for a given concrete web page. The reason for this restriction is that the estimated times for the parameters of the model are average values over a large sample of different pages. But, for example, the time required to detect the target link on a concrete page depends on the number of links on that page and more important on the visual structure of the page. Thus, the value of 9.6 s used in the model cannot be interpreted as an accurate estimation that is valid for each single page in the web.

However, the intention of the model is to compare mouse and keyboard navigation and to get a rough idea about the number of links a page can contain without causing a serious disadvantage for keyboard users. Both use cases are not influenced by the restriction described above.

### 4.2 An application of GOMS to investigate the efficiency of keyboard support in a web application

The model described in the previous section focuses on keyboard navigation of sighted users in Web sites. In order to address also web applications, some operators for data entry need to be considered.

Examples for such web applications are web shops, or Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) web applications used for professional work.

Compared to the model for keyboard navigation in web pages, in this case the time required to detect a target link on the screen and the probability of focus losses can be ignored. The reason is that web applications have a limited number of pages with a fixed structure. Expert users of such systems interact permanently with those pages and thus know the page structures very well. They have therefore not to search for a target on the screen, they simply know by heart where the target is located. Thus, simple KLMGOMS models are appropriate for this case.

As an example, a project is described in which GOMS was used during the early design phase of a CRM web application in order to evaluate if the planned keyboard support was sufficient. The pages in the web application contained many interactive elements, such as input fields, links, buttons, etc. Thus, it was clear that it was not sufficient to provide just standard keyboard support over the TAB-chain.

To increase the speed of keyboard navigation, a number of additional keyboard shortcuts were defined. To avoid conflicts with shortcuts used by assistive technology it was decided to allow the users personalize these shortcuts. As a side effect of this decision, the number of possible shortcuts was quite limited, since the shortcuts had to be implemented by scripting technologies. To guarantee performance and stability the amount of scripting in a page had to be limited.

Given these constraints, it was decided to implement eight navigational shortcuts, which allow the user to place the cursor focus to the first interactive element in eight defined screen areas. One of these screen areas (called work area) contained the main content of the page in the form of several groups of fields, tables or links. To allow fast navigation inside the work area, two additional shortcuts were implemented, which allow the user to move the cursor focus from within one group to the first interactive element in the next or previous group.

It was unclear if this quite restricted number of shortcuts allows efficient keyboard navigation. User performance is especially critical here, since the web application is used for professional work. First, we had to clarify what a sufficient keyboard support really means. The idea to quantify this concept is to compare keyboard and mouse navigation through a KLMGOMS analysis. It was decided to accept keyboard support as sufficient if the time to complete a set of standard tasks by keyboard only does not exceed 1.5 times[6] the time required to complete these standard tasks by mouse and keyboard.

[6] The value 1.5 was based on an analysis of typical user roles for the CRM web application. It was evident that if keyboard navigation is more than 1.5 times slower than mouse navigation, then it was unlikely that a user relying on keyboard navigation would be able to handle the typical workload without major problems.

A total of seven representative tasks were extracted. For each of these standard tasks a keystroke level GOMS model for pure keyboard interaction and a keystroke level GOMS model for the interaction with keyboard and mouse were elaborated.

The selected tasks are:

- navigate to a page starting from the entry page of the application,
- navigate from a search result list to the details page of an entry,
- navigate to a simple edit page to create an account that can be launched directly from the navigation menu,
- navigate to a complex edit page for account data that must be launched by first navigating to an account list and then by clicking on one of the list entries,
- create a task in the system,
- enter a sales order in the system,
- enter data to start a simple search request.

As an example of the KLMGOMS analysis, the first task and its representation in the models for mouse and keyboard navigation is described in detail.

To solve the first task using the mouse, the user has to move the cursor to an entry in the left navigation bar of the application and to click on an icon to open a menu. Then he or she has to click on one of the menu entries. After a server roundtrip (i.e., a system response time which is assumed to be 1 s) the target page opens. It is assumed that the hand is on the mouse when the user starts the sequence.

To solve the first task using the keyboard the user has to press a shortcut (ALT + 2) to place the cursor on the first entry in the navigation bar. Then he or she has to navigate to the correct link in the navigation bar by pressing the *Arrow Down* key six times. After that the user can open the menu with the *Arrow Left* key and can move to the correct menu entry by pressing three times *Arrow Down*. The entry can then be selected by pressing ENTER. Again, after a server roundtrip of 1 s, the target page appears.

The following abbreviations are used for the operators:

- P: Position the mouse cursor to a target on the screen (1.1 s),
- C: Mouse click (0.2 s),
- K: Press a key while typing a string (0.2 s),
- M: Mental preparation (1.35 s),
- S: Press a keyboard shortcut (0.54 s),
- T: Press a single navigation key, i.e., an arrow key or TAB (0.23 s),
- R: System response time (1 s).

With these abbreviations, the solution of the first task using the mouse can be represented as P C M P C R. Simply adding up the times for the single operators a value of 4.95 s is obtained.

**Table 1** Predicted GOMS times (in seconds) for the seven standard tasks

| Task | Keyboard only | Mouse + keyboard |
|---|---|---|
| 1 | 6.54 | 4.95 |
| 2 | 7.31 | 3.65 |
| 3 | 3.81 | 3.65 |
| 4 | 7.93 | 5.95 |
| 5 | 33.09 | 34.23 |
| 6 | 51.67 | 53.84 |
| 7 | 17 | 15.15 |

The solution of the first task by keyboard can be modeled as S M T T T T T T M T T T T R. Adding up the times for the single operators gives a value of 6.54 s.

Table 1 shows the estimated task completion times from the two KLMGOMS models (keyboard only and mouse and keyboard) for the seven selected standard tasks.

The average predicted GOMS times for the seven standard tasks show that the above stated goal for the keyboard support is met.

A quite simple KLMGOMS analysis was used to derive predictions concerning task completion times for mouse and keyboard navigation in the seven selected standard tasks. However, it is quite evident that absolutely accurate estimations cannot be expected from such an analysis.

For example, the analysis assumes that it takes 1.1 s to position the mouse cursor to a target on the screen. In a concrete situation, this time depends obviously to some extent on the length of the path between the actual cursor position and the target, and in addition on the size of the target. The value of 1.1 s is an average value calculated from observations over a wide range of different distances to the target and different target sizes.

Since the estimated times was used for the comparison of two different navigation strategies, it is not required that they reflect the exact user performance. To draw conclusions, it is sufficient that the ratios between the two navigation strategies are reflected. Assuming, for example, that users solve a task with mouse navigation twice as fast than with keyboard only navigation, then also the GOMS prediction for mouse navigation should be approximately twice as fast as the corresponding GOMS prediction for keyboard navigation.

An evaluation study performed on another web application indicates that this is indeed the case. The goal of this study was to compare the estimated times from KLMGOMS with actual user data.

For this study, four related tasks concerning the search functionality of the application were investigated:

- open a search page,

- enter a complex query in the search page and start the search,
- store the formulated query for later reuse,
- use a stored query by starting it directly from the entry page.

A GOMS analysis concerning mouse and keyboard navigation was done and the task completion times for these two navigation strategies were calculated.

Then the four tasks were investigated in a study with users. Four experienced computer users took part in this study. The tasks and especially available shortcuts for keyboard navigation were explained to them in detail in the instruction phase of the experiment. The participants had the chance to train on the tasks before the data collection starts.

Each participant had to solve each task three times using mouse and keyboard and in addition three times using keyboard only. Thus, we have per task and participant three measured task completion times for the interaction with mouse and keyboard and three measured task completion times for the interaction with keyboard only. All user interactions and the respective times were recorded by a tool installed on the test computer.

Table 2 shows the average task completion times (in seconds) of the participants and the predicted GOMS times for both navigation strategies. The last two columns show the ratio of the task completion times for mouse and keyboard navigation and the ratio of the GOMS predictions for these times.

The predicted task completion times from GOMS show for some of the tasks substantial deviations from the observed task completion times. Still, they reflect the ratios between the times observed for the two navigation strategies relatively accurately.

### 4.3 GOMS models for the interaction of blind users in web pages

The first two GOMS models presented describe the interaction of sighted users who are forced to navigate a web unit using the keyboard. These models are not adequate for the interaction of blind users with such applications. For blind users, it is important to consider the fact that they can scan the page only linearly over their screen reader or Braille display. Thus, the process to scan a page is much slower than for users who scan the pages visually. It also has to be considered that screen readers offer their users a number of features, for example fast skipping to the next element, the ability to show link lists, or the capabilities to search a page efficiently.

**Table 2** Comparison of the predicted task completion times from GOMS with observed data

| Task | Data | | GOMS | | Data | GOMS |
|---|---|---|---|---|---|---|
| | Keyboard | Mouse | Keyboard | Mouse | KB/ mouse | KB/ mouse |
| 1 | 5.16 | 2.94 | 5.19 | 3.65 | 1.76 | 1.42 |
| 2 | 40.91 | 31.94 | 37.55 | 30.65 | 1.28 | 1.23 |
| 3 | 9.53 | 10.62 | 9.93 | 11.15 | 0.90 | 0.89 |
| 4 | 6.79 | 3.44 | 6.86 | 3.9 | 1.97 | 1.76 |
| Overall | 62.39 | 48.94 | 59.53 | 49.35 | 1.27 | 1.21 |

A first approach to investigate efficiency-related problems of blind users in web pages is the calculation of the reaching time of a link [15, 16]. The reaching time of a link is defined as the average time a user working with a voice output needs to reach this link starting from the top of the page. Average reaching time is suggested in [15] as a measure for the navigability of a web page by blind users. Since there are many paths a blind user can use to reach a link reaching time is calculated from the shortest path to reach a link. For the calculation the heading structure, access keys or page internal navigation links are taken into account. The concrete calculation of the reaching time depends obviously on the speech rate of the voice output.

The calculation of the reaching time in [15, 16] depends on the length of the link texts on the shortest path from the top of the page to the target link. It considers the speech rate of the voice output, but no cognitive decision times or other user operations. Thus, the predicted times give only a very rough impression about the time needed by blind users to reach the links on a page. This rough and robust method to calculate reaching time is of course sufficient to detect basic problems with the structure of a page or the number of links in a page. But for an accurate prediction of the efficiency of navigation, more dimensions than the shortest path to the target and length of link texts must be considered.

A more sophisticated method to investigate efficiency-related problems of blind users can be found in [19, 20]. These papers describe a NGOMSL model[7] for the navigation behavior of blind users in web pages.

The model was derived from user studies and addresses the following aspects of blind users' web navigation (for details see [19]):

- verification of a spoken text,

[7] In fact this model is not a plain NGOMSL model, since it enhances GOMS with an additional mechanism to describe the special behavior of screen reader users. A user of a screen reader has in principle three strategies to find an element on a web page: searching for the element in question, using the list mode of the screen reader, or using the skip key to quickly step over irrelevant elements. If one of these options fails, the user will try another one. Selection rules are not able to describe such an interaction behavior properly [19]. Therefore, an additional mechanism, called a *choice* was established. In this context, a choice is a set of equal alternatives to reach a goal.

- page internal navigation with quick key, over the list mode, and with the help of the search function,
- choices between the alternative navigation strategies (quick key, list mode, search),
- listening to content over the speech output,
- reading content over the Braille display, etc.

This model has the potential to make much more accurate predictions than the simple calculation of reaching time. Currently, the model is not yet validated through a user study [19] and some of the times for basic operators are unknown. This makes the concrete application of the model difficult for the time being.

## 5 Conclusions

Accessibility guidelines are extremely important to show basic problems which must be avoided to design accessible web units. However, designers must be aware of the fact that guideline compliance is not sufficient to ensure practical usability and accessibility for disabled users. As shown by some of the examples presented in this paper, efficiency-related issues with the design of a web unit can be as exclusive for disabled users as violations of basic guidelines. Thus, designers of web units need tools to detect and analyze such efficiency relevant design issues.

GOMS analysis is an established method to develop insights into the efficiency of design alternatives. It allows estimating the average task completion time by trained computer users. This paper has shown that it is possible to formulate GOMS models, which describe the interaction behavior of disabled users. Thus, GOMS analysis can be used as a tool to investigate efficiency-related design issues for disabled users.

GOMS analysis has two important advantages. First, it does not require user participation and is thus a quite inexpensive method of interface evaluation. Second, it does not require a running system or even a high fidelity prototype. Therefore, it can be applied early in the design phase to investigate potential efficiency-related issues for disabled users. Overall, GOMS analysis or more general engineering models of human performance are valuable tools to investigate the efficiency of interface designs for different groups of disabled users.

Obviously, GOMS analysis is restricted to the prediction of efficiency problems. For example, the question of how easy it is to understand the content of a Web site for blind users cannot be answered by a GOMS analysis.

The average web designer cannot be expected to perform a GOMS analysis, since this is technically not trivial and also time consuming. Thus, it is important to include checks concerning efficiency-related accessibility problems into automatic check tools. GOMS models are very promising concerning this aspect, since they allow calculating basic task completion times on the basis of an analysis of the structure of a web unit.

A nice example of how such a tool can look like is aDesigner [16]. This tool simulates the interaction of blind or visually impaired users with Web sites. It includes an analysis of efficiency-related issues, for example reaching times[8] for links on a web page. When a web page is analyzed with aDesigner, the system calculates for each link in the page the average time a user working with a voice output needs to reach this link starting from the top of the page. These times are then visualized by color gradients on the original page. A warning is given if the reaching times for some links exceed a given threshold.

It is extremely easy to analyze potential problems with such a tool, since the web designer does not need a deeper understanding into the method used to calculate the reaching time. In addition, the efficiency of implemented shortcuts or a better structuring of the page content on the reaching times of the links in the page can easily be verified. It would be a big step forward if other GOMS models describing the interaction behavior of disabled users would be implemented in a similar way.

This paper has described a model for the selection of a target link in a web page by sighted users who are forced to navigate a web unit by keyboard. In addition, it has described the application of a KLMGOMS analysis to investigate the efficiency of keyboard support in a web application used for professional work. These examples show the potential of GOMS models, or of more general engineering models of human performance, to make quantitative predictions concerning the efficiency of disabled users. But these models cover only some situations of interest. Further models of this type need to be developed to get more insights into efficiency problems for disabled users.

## References

1. Stephanidis, C., Salvendy, G.: Towards an information society for all: HCI challenges and R&D recommendations. Int. J. Hum. Comput. Interact. **11**(1), 1–28 (1999)
2. Bühler, C., Stephanidis, C.: European co-operation activities promoting design for all in information society technologies. In: Miesenberger, K., Klaus, J., Zagler, W., Burger, D. (eds.) ICCHP—Computer Helping People with Special Needs. Lecture Notes in Computer Science, vol. 3118, pp. 80–87 (2004)

[8] Please note that the calculation of reaching time in aDesigner is currently based on a simple algorithm and not on a GOMS model.

3. Caldwell, B., Cooper, M., Reid, L.G., Vanderheiden, G.: Web content accessibility guidelines 2.0. W3C Working Draft. http://www.w3.org/TR/WCAG20/ (2007). Accessed 17 May 2007
4. US Department of Justice. Sect. 508 of the Federal Rehabilitation Act. http://www.section508.gov/
5. Chrisholm, W., Vanderheiden, G., Jacobs, I.: Web content accessibility guidelines 1.0. http://www.w3c.org/TR/WCAG10/ (1999)
6. Mohamad, Y.; Stegemann, D.; Koch, J., Velasco, C.A.: Imergo: supporting accessibility and web standards to meet the needs of the industry via process-oriented software tools. In: Miesenberger, K., Klaus, J., Zagler, W., Burger, D. (eds.) ICCHP—Computer Helping People with Special Needs. Lecture Notes in Computer Science, vol. 3118, pp. 310–316 (2004)
7. Bühler, C., Heck, H., Nietzio, A., Craven, J., Snaprud, M.: Combining empirical and theoretical methods to enhance large scale web accessibility monitoring results. In: Eizmendi, G., Azkoitia, J.M., Craddock, G.M. (eds.) Challenges for Assistive Technology. IOS Press, Amsterdam (2007)
8. Watchfire: Welcome to Bobby WorldWide. http://bobby.watchfire.com/bobby/html/en/index.jsp (2004)
9. Petrie, H., Kheir, O.: The relationship between accessibility and usability of Websites. In: CHI 2007 Proceedings (2007)
10. Adams, R., Gill, S.: User modelling and social intelligence. In: Stephanidis, C. (ed.) Universal Access in Human Computer Interaction. Coping with Diversity. Lecture Notes in Computer Science, vol. 4554, pp. 584–592 (2007)
11. Antona, M.; Mourouzis, A., Stephanidis, C.: Towards a walk-through method for universal access evaluation. In: Stephanidis, C. (ed.) Universal Access in Human Computer Interaction. Coping with Diversity. Lecture Notes in Computer Science, vol. 4554, pp. 325–334 (2007)
12. Carey, K., Garcia, R., Power, C., Petrie, H., Stefan Carmien, S.: Determining accessibility needs through user goals. In: Stephanidis, C. (ed.), Universal Access in Human Computer Interaction. Coping with Diversity. Lecture Notes in Computer Science, vol. 4554, pp. 28–35 (2007)
13. International Standards Organization: Standard 9241: Ergonomic requirements for office work with visual display terminals (1992–2000)
14. Disability Rights Commission: The Web Access and Inclusion for Disabled People. A formal investigation conducted by the Disability Rights Commission. TSO, London (2004)
15. Fukuda, K., Saito, S., Takagi, H., Asakawa, C.: Proposing new metrics to evaluate web usability for the blind. In: CHI 05: Extended abstracts on Human Factors in Computing Systems, pp. 1387–1390. ACM Press, New York (2005)
16. Takagi, H., Asakawa, C., Fukuda, K., Maeda, J.: Accessibility designer: visualizing usability for the blind. In: Assets 04: Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 177–184. ACM Press, New York (2004)
17. King, A., Evans, D.G., Blenkhorn, P.: The evaluation of a web browser for blind people. In: Pruski, A., Knops, H. (eds.) Assistive Technology: From Virtuality to Reality, pp. 637–641. IOS Press, Amsterdam (2005)
18. Schrepp, M.: On the efficiency of keyboard navigation in Web sites. Universal Access in the Information Society, vol. 5, no. 2, pp. 180–188 (2006)
19. Tonn-Eichstädt, H.: Measuring Website usability for visually impaired people—a modified GOMS analysis. In: ACM SIGACCESS Conference on Assistive Technology, pp. 55–62. ACM Press, New York (2006)
20. Eichstädt, H.: Interaktionen blinder Nutzer bei der Bedienung linearisierter Oberflächen (Interactions of blind persons during the usage of linearized user interfaces). In: Stary, C. (ed.) Mensch and Computer 2005: Kunst und Wissenschaft—Grenzüberschreitungen der interaktiven ART, pp. 61–70. Oldenbourg Verlag, Munich (2005)
21. Card, S.K., Moran, T.P., Newell, A.: The Psychology of Human–Computer Interaction. Erlbaum, Hillsdale (1983)
22. John, B.E., Kieras, D.E.: The GOMS family of user interface analysis techniques: comparison and contrast. ACM Trans. Comput. Hum. Interact. **3**(4), 320–351 (1996)
23. Kieras, D., John, B.: Using GOMS for user interface design and evaluation: which technique? ACM Trans. Comput. Hum. Interact. (TOCHI) **3**(4), 287–319 (1996)
24. Olson, J.R., Olson, G.M.: The growth of cognitive modelling in human–computer interactions since GOMS. Hum. Comput. Interact. **5**, 221–265 (1990)
25. Kieras, D.: A guide to GOMS model usability evaluation using NGOMSL. In: Helander, M., Landauer, T.K., Pradhu, P. (eds.) Handbook of Human–Computer Interactions. Elsevier, Amsterdam (1997)
26. Schrepp, M., Fischer, P.: GOMS models to evaluate the efficiency of keyboard navigation in web units. Eminds Int. J. Hum. Comput. Interact. **I**(2), 33–46 (2007)
27. Gray, W.D., John, B.E., Atwood, M.E.: Project Ernestine: validating a GOMS analysis for predicting and explaining real-world task performance. Hum. Comput. Interact. **8**, 237–309 (1993)
28. Keates, S., Clarkson, P.J., Robinson, P.: Developing a methodology for the design of accessible interfaces. In: Proceedings of the 4'th ERCIM workshop. Stockholm, Sweden (1998)
29. Schrepp, M., Fischer, P.: A GOMS model for keyboard navigation in web pages and web applications. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) Computers Helping People with Special Needs. 10th International Conference, ICCHP 2006. Linz, Austria, July 2006. Lecture Notes in Computer Science, vol. 4061, pp. 287–294 (2006)