Using Communicability Evaluation to Compare Interaction Design of Two HTML Tag Editors

Overview

The focus of this study is on the use of communicability evaluation to analyze the user's experience with two instances of the same type of application—HTML tag editors. *Communicability is the distinctive quality of interactive computer-based systems that communicate efficiently and effectively to uses their underlying design intent and interactive principles*. Our purpose here is to show how the results of this particular evaluation method can identify, explain, and inform the redesign of problematic interaction design.

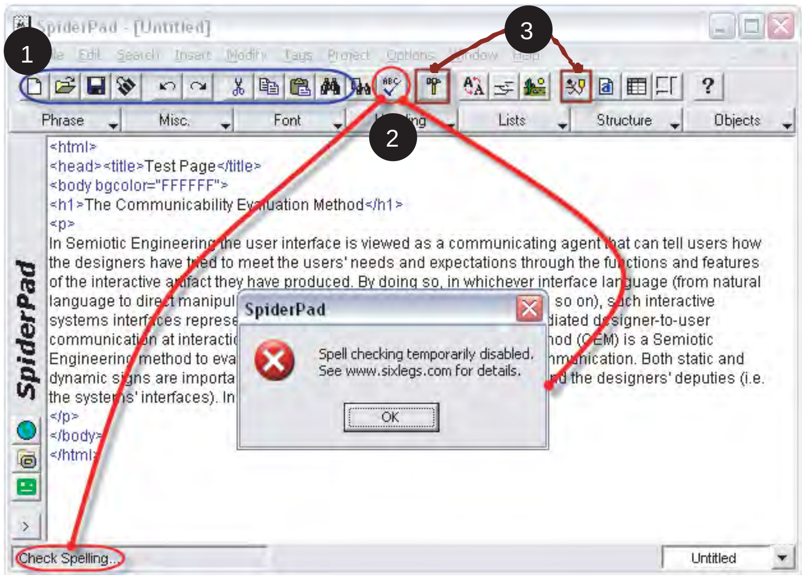
We will start by briefly describing the method. Then we will present the case study. We will provide links to further material, discussing other situations where communicability evaluation can be used, and to what effects

The Communicability Evaluation Method

In Semiotic Engineering the user interface is viewed as a communicating agent that can tell users how the designers have tried to meet the users' needs and expectations through the functions and features of the interactive artifact they have produced. By doing so, in whichever interface language (from natural language to direct manipulation, from gestures to voice commands, and so on), such interactive systems interfaces represent the designers and allow for computer-mediated designer-to-user communication at interaction time. This is why we say that such interfaces are *the designers' deputies*—they speak for the designers.

The Communicability Evaluation Method (CEM) is a Semiotic Engineering method to evaluate the quality of such designer-to-user communication. Both static and dynamic signs are important for mutual understanding between users and the designers' deputies. In Figure 1, for example, we see a static picture of an interactive sate in SpiderPad, with some indications of the dynamics of interaction.

Static signs (e.g. icons on the tool bar) communicate to users (on the behalf of the designer) what the users can do with the system. The icons surrounded by a blue box belong to a *graphic*



```
<html>
<head><title>Test Page</title>
<body bgcolor="FFFFFF">
<h1>The Communicability Evaluation Method</h1>
<p>
In Semiotic Engineering the user interface is viewed as a communicating agent that can tell users how
the designers have tried to meet the users' needs and expectations through the functions and features
of the interactive artifact they have produced. By doing so, in whichever interface language (from natural
language to direct manipul                                 so on), such interactive
systems interfaces represe                                 diated designer-to-user
communication at interacti                                 hod (CEM) is a Semiotic
Engineering method to eva                                  munication. Both static and
dynamic signs are importa                                  d the designers' deputies (i.e.
the systems' interfaces). In
</p>
</body>
</html>
```

SpiderPad

Spell checking temporarily disabled.
See www.sixlegs.com for details.

OK

Figure 1 *Communicating design through SpiderPad's interface*

language used in numerous editors designed for Windows. Among the things they *tell* users are the following: (a) *''Press here to* {*create,open,save*} *a file''; (b) ''Press here to* {*redo,undo*} *the last thing you've done''; (c) ''Press here to* {*copy,paste*} *objects* {*to,from*} *the clipboard''*; etc. Together, these icons tell users things like: *''With this system you can: create, open, and save file; copy, paste, cut, and search text;* {*etc*}*.''* Because of certain visual patterns, they also communicate that this is a system designed for Windows (and not for Unix, for example).

The icons surrounded by a burgundy box are more rarely encountered. In fact, the one to the right is possibly not encountered outside SpiderPad. The icon, for instance, is meant to communicate that if users press that button they will be able to edit the attributes of the {*selected,surrounding*} tag, whereas the icon is meant to communicate that if users press that button they will be able to edit the *body* tag. Note that, at first sight, the meaning of these two instances of designer-to-user communication is less obvious to users, although to English-speaking users the meaning of becomes memorable once the play on words gets across.

Dynamic signs (i.e. those that appear in the process of interaction) also convey the designers' message to users. In Figure 1 we see the sketch of a short span of interaction. When the user passes the mouse over the button, the system (speaking for the designer) sends the users a message saying that it activates the "Check Spelling" function. When he/she clicks on that button, another message is sent saying that the user's goal was not achieved (" ") because "Spell checking is temporarily disabled [and the user should] see `www.sixlegs.com` for details". As users *unfold* all such interactive signs while using SpiderPad, they gradually make sense of what SpiderPad means.

The result of such sense making, in Semiotic Engineering, is the user's interpretation of SpiderPad. CEM is therefore a method to evaluate how well designers communicate their design intent to users. It does so by capturing, categorizing, interpreting, and analyzing *communicative breakdowns* in test situations. Such communicative breakdowns help evaluators identify, explain, and inform the redesign of problematic interaction design.

Steps in Communicability Evaluation

CEM is achieved in three main steps: *tagging*, *interpretation*, and *semiotic profiling*. Prior to these, there are two subsidiary steps: *test preparation* and *test application*.

Test Preparation

CEM preparation includes the following activities:

1. A thorough study of online and offline documentation of the application.
2. A thorough inspection of the application. (It is a good idea to talk with the designers about the design intent, if this is possible.)
3. The selection of test scenarios and activities, based on critical design issues identified after reading the documentation, inspecting the application, and talking to the designers. From a Semiotic Engineering perspective, a critical design issue is one where the designer-to-user communication is likely to fail. When design alternatives are being compared, the alternatives themselves constitute critical issues.
4. The selection of test participants. The number of participants is typically small. CEM is not a quantitative method, but a qualitative one. Results do not depend on the number of tests, but on the depth and extent of semiotic associations that evaluators can established between *intended signs* (obtained from reading documentation, inspecting the application, and talking to designers) and *interpreted signs* (obtained from observing)
5. interaction and interviewing participants).
6. Preparing interviews. Knowing the participants is important to help evaluators detect how users are likely to *receive* the designer's communication, how familiar they are with the type of interface language they will have to use in order to communicate with the application, what general expectations they have regarding the use of computer tools to achieve the activities they are about to perform, and so on.
7. Preparing printed material (if there is any). This may include a description of activities, forms and/or questionnaires.
8. Setting up the equipment and the application. Tests must be recorded. Whenever possible, the interaction with the application should be logged with screen capture software. The use of video tape may complement screen capture. The advantage of using both is to be able to see the participants' body language, and capture verbalizations most of them produce during the test. The meanings conveyed by body language and verbalizations, synchronized with interactions,

provides additional semiotic evidence of the presence of certain interpretive signs like ‘‘I don’t know what to do now’’ (a blank look, no body movement), or ‘‘This is getting on my nerves’’ (an angry look, the participant vociferates at the computer). Don’t forget to check the quality of screen captures and video tape *before* tests are run—frame rates for screen capture and lighting for video tape may ruin a test. Moreover, since some tests are likely to take some time, it is a good idea to make sure that there is enough disk space, especially for temporary files. It is also desirable to have a clone (or a TV) monitor in a separate place, where an observer can closely follow the participant’s interaction and prepare questions for a post-test interview (see below).

Test Application

CEM application includes the following activities:

1. Welcoming and interviewing participants, prior to the interactive session. This includes a warm up conversation, an explanation of the test and its purposes (in general—at times, detailed explanations of purposes may influence results), and a pre-test interview.
2. Helping users feel comfortable with the equipment and the system (e.g. checking that the physical conditions of the setting are OK, that the participants are relaxed and in a good mood).
3. Checking that all recording equipment is ready to go, and that the tester (evaluator) can take notes during the test.
4. Taking notes during the test. The evaluator should be thoroughly familiar with CEM, and try to detect communicability breakdowns *on the fly* during the test. He should note if there are ambiguous symptoms of how the user has interpreted the designer’s communication through the interface (e.g. if the participant achieves a task following an *unexpected* sequence of operations, is it because she does not get the design message or because she deliberately chooses to do something else?). Interviewing the participant after the test, and asking her what she had in mind during interaction helps increase the quality of results. Also, asking what participants thought of the experience, what impressions they got from the application, and so on, provides a wealth of material for semiotic analyses.
5. Checking that recordings are safely stored as soon as the test is over. Screen capture movies may be quite large files, depending on the data compression supported by the screen capture software being used.

Tagging

Tagging amounts to annotating screen capture movies with *communicability evaluation tags*. These are stereotypical expressions that the evaluator puts in the participant’s mouth, like: *‘‘What’s this?’’*, *‘‘Where is it?’’*, *‘‘Where am I?’’*, *‘‘Help!’’*, etc. Each expression is associated to patterns of breakdown in communication. The full set of tags used in CEM is presented in [Table 1].

Tag	*Symptom*	*Remarks*
Where is it?	The user knows what she is trying to do (to *communicate*) but cannot find an interface element that will *tell* the system to do it. She typically browses menus, opens and closes dialog boxes, hides and *un*hides window elements, looking for that particular element.	Searches for interface activation elements usually start by *an educated guess* of where the element is most likely to be placed. However, after some time there may be no more guesses to make, and the search may turn into a ''raster scanning'' of the interactive space. If the user finds the element as a result of *an educated guess* the breakdown is less serious than if the element is only found after a long scanning of the interface.
What now?	The user doesn't know what to do next (what to tell the system), and so she wanders around the interactive space searching for an opportunistic clue to restore productive communication with the system. Menus, dialog boxes, and toolbars may be inspected at random, or in sequence.	The symptoms associated to ''Where is it?'' and ''What now?'' may be virtually the same. The difference between the two breakdowns is that in one case the user knows what she is doing, and in the other she doesn't. It is most often impossible to tell, from the screen capture only, if the user knows or doesn't know what she is doing. Therefore, the symptoms associated to ''Where is it?'' and ''What's this?'' are usually a topic for disambiguation during the post-test interview.
What's this?	The user does not understand an interface sign, and looks for clarification. She may pass the mouse over the element in order to get a tool tip, or examine the behavior of the interface sign (to see how the system responds).	Inspecting the meaning of interface elements may come in association with other breakdowns, or not. For instance, the user may be engaged in productive interaction (no breakdowns) and suddenly take a side path to *ask the system* what a particular element means. Alternatively, the user may *ask the system* for clarifications while

(*continued*)

Table 1 *Communicability Evaluation Tags*

Tag	*Symptom*	*Remarks*
		trying to resolve a breakdown. For example, the inspection of menus, dialog boxes, and other interface elements while trying to find the answer to ''Where is it?'' or ''What now?'' may involve one or more ''What's this?''
Oops!	The user makes an instant mistake in interaction, and immediately corrects herself. A typical symptom of ''*Oops!*'' is to undo the faulty operation triggered by miscommunication.	The correction of certain operations is easily done, by means of the *undo* function. Other operations cannot be *undone*. In the latter case, the user may have to plan how to restore the state of the system prior to the miscommunication. This may be a short path, or end up into a major breakdown, from which the user will recover, or not. (See tags ''I give up.'' and ''Looks fine to me.'')
Where am I?	The user is telling things to the system that would be appropriate in another context of communication. This may result from a misinterpretation of the current interactive context. Symptoms may include trying to select objects that are not active in the current context, or trying to interact with signs that are *output only,* for example.	There are usually remarkable similarities between the current context of interaction (where the user's communication is ineffective) and *another* context of interaction (where the same communication *would be* effective). Users of applications with *preview* functions and *WYSIWIG* object editing style are often confused, and try to edit objects while previewing how they will be printed.
I can't do it this way.	While trying to achieve a goal or sub-goal, the user engages in a several-step sequence of operations, but suddenly realizes that this is not the right thing to do. So, she abandons that sequence, and takes another path.	The difference between *''Oops!''* and *''I can't do it this way.''* is the range of equivocal actions communicated to the system. *''Oops!''* characterizes a single action, which is instantly revoked. *''I can't do it this way.''* involves a longer sequence of

Table 1 *(continued)*

Tag	*Symptom*	*Remarks*
		actions, which are abandoned for another path.
Why doesn't it?	The user insists on repeating a certain operation that does not produce the expected effects. The user is aware that these effects are not produced, and that others are produced instead. But insists on doing the same thing time and again, because she doesn't understand why the interaction is not right. The repetition of the operation may be in sequence, or separated by one or more different operations.	The typical reason for a *''Why doesn't it?''* is that the user strongly believes that what she is doing should cause the desired effects. So, she insists on the same kind of action, trying to find out if a particular change of parameters or context is possible or needed to make the action ''work''. Occasionally, the user may *anticipate* that the effects will not be what she expects, and not fully activate the function. Nevertheless she will start the same communication over again (like activate the same dialogue box, open the same sub-menu, etc), and abandon it before its full achievement.
What happened?	The user fails to understand the system's response to what she told it to do. The typical symptom of *''what happened?''* is the repetitive activation of an operation whose effect is absent, or not perceived.	Repetitive actions can be tagged with *''Why doesn't it?''* or *''What happened?''* The difference between the two is that in the first case the user can see the effects caused by what she is telling the system, whereas in the latter she can't. Occasionally, *''What happened?''* may be associated to a single activation of a function, followed by another action that clearly indicates that the user failed to get the system's response message. For example, the system may have given indication that the user's goal was achieved, or cannot (*continued*)

Table 1 *(continued)*

Tag	*Symptom*	*Remarks*
		be achieved for some overriding reason, or else that it can be achieved if some particular path is followed. Because she did not get this message, her follow-up action may be totally inconsistent with the system's communication.
Looks fine to me.	The user believes she has achieved her goal, although she hasn't. The typical symptom is when the user declares she has successfully finished a task, when she hasn't, really.	*''Looks fine to me''* is used to qualify the user's attitude towards the final state of a task or sub-task. It is important to have evidence that the user is indeed satisfied with the results, and not simply abandoning the task shortly before finishing it.
I give up.	The user believes that she can't achieve her goal, and interrupts the interaction (communication) with the system.	Like *''Looks fine to me''*, *''I give up''* is used to indicate the failure in achieving a proposed task. However, the user's attitude in one case is the opposite of the other. *''I give up''* Indicates the user's conscious frustration and recognition that she could not tell the system to do what she wanted (possibly because she could not make sense of what the system was telling her in the first place).
I can do otherwise.	Because she can't see or understand that the system is telling her what she can or must say to achieve a particular goal, the user communicates her intention by means of unexpected signs. This may include achieving a goal because of the (side) effects of interaction meant for other purposes. However, it most often corresponds to a sub-optimal way of communica-	It is crucially important that the user not be aware of better solutions communicated by the system. Typically, there are some previous breakdowns in user-system communication when *''I can do otherwise''* finally happens. However, although less usual, this tag may be the first to appear in an interactive session. The evaluator should clarify the user's presuppositions about how the

Table 1 *(continued)*

Tag	*Symptom*	*Remarks*
	tion (e.g. taking longer paths or adopting a solution that is faulty in some respects, in spite of the system's communication of better solution alternatives through the interface).	system works during the post-test interview.
Thanks, but no, thanks.	Although the user has understood the system's communication about which design solutions should or are expected to be preferred to others, she deliberately chooses to communicate her intent with unexpected signs.	As with other tags, *''Thanks, but no, thanks''* is associated to the same kinds of symptoms as another tag, namely *''I can do otherwise.''* The difference lies in the user's attitude. Hence the importance of post-test interviews. Although the user *understands* the system communication in this case, it is nevertheless a case of communicative breakdown because the designer's deputy discourse will always communicate a particular design intent by means of signs that the user considers less adequate for the context than the ones she chooses to use.
Help!	The user explicitly asks for help, either the system's, or somebody else's (e.g. the evaluator's). Accessing online help, searching online or offline documentation, talking to others, online or offline, about the problem, all constitute symptoms of this kind of communicative breakdown.	*''Help!''* is a kind of meta-level tag, in that it projects the user into explicit communication about communication with the system (or *metacommunication*, technically). This is not necessarily a serious breakdown—in fact, the evaluator may realize that this is *not* a breakdown instance at all. It may well be a sign of the user's curiosity about the system, which ultimately amounts to successful interaction, rather than unsuccessful one. The difference between one case and the other should be clarified during the post-test interview.

Table 1 *(continued)*

Interpretation

Interpretation, as the name suggests, amounts to answering the following question: *''What **meanings** does the evaluator assign to the tagged movies?''* The answer to this question will tell how successful the designer's communication is. Success is associated to the absence (or insignificant amount) of communicative breakdowns.

The evaluator should pay attention to such factors as:

1. How often, and in which particular context, each type of tag has appeared (per participant, per task, overall)?
2. Are there tagging patterns (similar sequences of tags) that can be identified (across tasks for the same participant, across participants for the same task, overall)?
3. Can tag types or sequences be regularly associated to problems in establishing communicative goals or sub-goals?
4. If (as is desirable) the evaluator has used additional evaluation methods (e.g. heuristic evaluation or a cognitive walkthrough), is there a correspondence between the *locus* of tag occurrence and that of problems indicated by the other methods?

Answers to the questions above will help the evaluator ***interpret*** the meaning of tagged movies and decide if there are communicability problems with the application under evaluation. If there are, the evaluator will be able to tell *what* problems, and *why*. The explanation is referred to one or more of the following categories of breakdown in communication:

- The user cannot *express* what he/she means.
- The user chooses the *wrong way to express* what he/she means.
- The user cannot *interpret* what the system expresses.
- The user chooses the *wrong interpretation for* what the system expresses.
- The user cannot even *formulate a communicative intent*.

All of the above turn around the essence of communication: expressing content and intent, by using signs at hand. These signs may spring from a variety of origins, like: sign systems deeply ingrained in the culture (or sub-culture) of all interlocutors involved in communication; sign systems from a culture (or sub-culture) that not all interlocutors share; sign collections (not necessarily *systems*) associated to the context of communication; and even signs incidentally invented or transformed (as is the case in metaphoric communication, for example) by interlocutors.

Semiotic Profiling

The semiotic profiling step should help the evaluator identify, explain, and inform the redesign of problematic interaction design. This is achieved though a *reconstruction* of the designer-to-user global communication message. The content of the message can be summarized as this (the first person "*I*" in the message is the designer, or a spokesperson for the design team):

> "Here is my understanding of who you are, what I've learned you want or need to do, in which preferred ways, and why. This is the system that I have therefore designed for you, and this is the way you can or should use it in order to fulfill a range of purposes that fall within this vision."

Tagged interactions, and their corresponding interpretations, will allow the evaluator to find evidence to answer the following questions:

1. Who do the designers think are the users of the product of their design? (i.e. Who are the apparent receivers of the designers' communication?)
2. What do they think are these users' wants and needs? (i.e. How is communication tailored to privilege certain wants and needs, and not others?)
3. Which do they think are these users' preferences with respect to their wants and needs, and why? (i.e. How and why is the *users'* communication with the system facilitated in certain contexts, and not others? Are there communicative choices available?)
4. What system have they *consequently* designed for these users, and how can or should they use it? (i.e. How consistently do the design principles match the designers' communication about the users they expect to be talking to?)
5. What is the designers' *global design vision*? (i.e. How is the technology made available through the system expected to be received, and to transform the users' lives in a positive way?)

If tagging and interpretation provide evidence of communicative problems, questions 1–5 will lead the evaluator sense making, and help identify the reasons for such problems. All signs involved in communication and miscommunication constitute a *trace* of the user's interpretation and/or intent while interacting with the system. One of the advantages of CEM is that such traces can inform a *redesign* of the application. For example, a high frequency of "Where is it?" tags may be associated to some patterns of search—suppose that users have difficulty to find *where* some configuration tool *is*, and that most users *start* looking for it in the "Tools" menu. They do not find it there, because the designers decided that the configuration tool should be grouped with the options of some other menu (e.g. "View"). The tagged movies are telling evaluators that the users' understanding of that particular *configuration* activity is related to their understanding of what *tools* can be used for (and not what *view* is all about).

The comparison between the design *intent*, as witnessed by the creators of the application, and the design *interpretation*, as witnessed by the interactive sessions with users, is guided by tagging and interpretation. And the conclusion of the comparison (plus explanations and suggestions) is achieved in the semiotic profiling. As a result, CEM is a fine characterization of the quality of the user-system communication, in view of the *purpose* of design. **Good communicability** means that designers have got their message across very well. Users may, however, *repurpose* the application, and happily use the technology in some unanticipated way. This is an interesting situation (not unusual for extensive parts of the technology, if not for the technology as a whole), which other evaluation methods don't pay much attention to. If users got the designers' message, alright, but decided to use the technology in some other way ("Thanks, but no, thanks."), communicability was good. However, if the users did not get the designers message right, and imagined some other interpretation for the technology that nevertheless allows them to use it for their purposes ("I can do otherwise."), *there is* a communicability problem with the application. The problem may be serious if a usability problem is caused by the misunderstanding—for example, if users do what they want to do but spend much more time and effort than they would if they had got the designers' message right. This situation shows the explanatory power of CEM—it is not a *predictive* explanation, but a *descriptive* one.

The Case Study

Because of the explanations provided above, we will only briefly describe the essence of CEM steps in this study.

Arachnophilia and SpiderPad

The study involved the use of two HTML tag editors: Arachnophilia 3.9 and SpiderPad 1.5.3. Both editors can be downloaded from the Web.

Arachnophilia [`http://www.arachnoid.com/arachnophilia/`] allows users to:

- Create HTML pages using a suite of powerful tools.
- Upload your Web site to your Internet service provider using Arachnophilia's built-in FTP service.
- Fully customize Arachnophilia's menus, toolbars and keyboard commands. Arachnophilia lets you create or remove any commands, toolbars, or menus you want to.
- Beautify, and analyze the structure of, your Web pages, so they will be more likely to be error-free and work correctly with more browsers.
- Create working environments for many kinds of programming tasks using Arachnophilia's fully customizable menus and toolbars.

SpiderPad [`http://www.sixlegs.com/`] allows users to:

- Create/edit HTML code
- Create templates which will prompt you for information when used
- Edit/Correct tag attributes
- Customize the interface (and application) for greater efficiency and comfort
- Use graphic tools to design tables, frames and forms

Purpose of the Evaluation

The practical purpose of the evaluation was to compare how users interpreted the designer's message in one case and the other. Although both editors have been designed to help users create and edit HTML pages, the peculiarities of design are quite different. So, the idea was to investigate how well these peculiarities were communicated to users.

Participants

Ten participants with varying levels of expertise in HTML were selected. None of them had had any previous contact with either editor. Participants were split up into two groups: one worked first with Arachnophilia, and then with SpiderPad. The other followed the reverse order. After 5 minutes of exploration with each editor, they were asked to: (1) create a nested list of items, with particular numbering and bullet types; (2) change the background color of an existing HTML page; (3a) create a 2x2 table with border and title (caption); and (3b) edit the table by merging the two cells in the first row. They were allowed a maximum of 4 minutes to perform each task.

The Physical Setup

Tests were carried out in a lab situation. Participants used a personal computer. The session was logged with Lotus ScreenCam, and videotaped. An observer took notes and carried out pre- and post-test interviews.

An example of the Participants' Activities

In the first task, users were asked to create a white page, with two lists: a numbered list with 2 items, and an indented bulleted list of 3 items (all items are words in Portuguese):

1. arroz
2. frutas
    - abacaxi
    - uva
    - banana

Two examples of the test records

The following is a description of movies *Task 1A Arachnophilia* and *Task 1A SpiderPad*. (If needed, download *CODEC file* for playback with Windows Media Player.)

User's interaction with Arachnophilia:

> The user first creates a new page (File > New file > HTML file). He sets the background and text colors as white and black, respectively. Next, he locates the TITLE tag and types in the title of the web page. He opens the Struct and Styles toolbars, possibly looking for a list wizard. For the numbered list, instead of using specific HTML tags, he types in the numbers, dots, and items. For the bulleted list, he browses the toolbar buttons, possibly looking for a list wizard. He shows the ''Struct'' toolbar and opens the List Wizard dialog box, hesitates hovering between the Create and Hide buttons, and finally dismisses the dialog box by clicking on the ''X'' close button located at the top-right corner of the box. He then clicks on the LI button on the Struct toolbar, and the application inserts an <LI> tag on the document window. He then types the first item of the bulleted list, followed by carriage return. He types the remaining items without the <LI> tag. He hovers over a few toolbar buttons, and returns to the LI button. He moves the cursor to the beginning of the second list item, and clicks on LI. The application then inserts the second <LI> tag. The user then moves to the beginning of the third list item, and clicks on LI again, to insert the third <LI> tag. He starts to browse the Struct toolbar buttons again, possibly looking for a way to indent the whole list. He shows the Graphics toolbar, shows and hides the Forms toolbar. He goes back to the LI button, hesitates over it but presses the neighboring Bot button, which inserts an application-specific tag. He deletes the tag, hesitates a little bit, moves the window, scrolls the document up and down, and declares that he has completed the task.

However, the user didn't succeed in creating the intended web page, because the resulting bulleted list was not indented with relation to the first.

When reviewing the interaction movie, the evaluators *tagged* it, i.e. identified moments of interaction breakdown with communicability tags. We now repeat the narration, indicating in boldface the tags that the evaluators assigned to segments of the movie, formatted in underlined text.

> The user first creates a new page (File > New file > HTML file). He sets the background and text colors as white and black, respectively. Next, he locates the TITLE tag and types in the title of the web page.
>
> *He opens the Struct and Styles toolbars, possibly looking for a list wizard.* [**WHERE IS IT?**]
>
> *For the numbered list, instead of using specific HTML tags, he types in the numbers, dots, and items.* [**I CAN DO OTHERWISE**]
>
> *For the bulleted list, he browses the toolbar buttons,* [**WHAT'S THIS?**]
>
> *again possibly looking for a list wizard.* [**WHERE IS IT?**]
>
> He shows the ''Font'' toolbar but opens the List Wizard dialog box, from the Struct toolbar.
>
> *He hesitates hovering between the Create and Hide buttons,* [**WHAT'S THIS?**]

possibly looking for a way to dismiss the dialog box. [**WHERE IS IT?**]

Finally, he dismisses the dialog box by clicking on the ‘‘X’’ close button located at the top-right corner of the box. [**I CAN’T DO IT THIS WAY**]

He then clicks on the LI button on the Struct toolbar, and the application inserts an <LI> tag on the document window. He then types the first item of the bulleted list, followed by carriage return. He types the remaining items without the <LI> tag. He hovers over a few toolbar buttons, and returns to the LI button. He moves the cursor to the beginning of the second list item, and clicks on LI. The application then inserts the second <LI> tag. The user then moves to the beginning of the third list item, and clicks on LI again, to insert the third <LI> tag.

He starts to browse the Struct toolbar buttons again, [**WHAT’S THIS?**]

possibly looking for a way to indent the whole list. He shows the Graphics toolbar, shows and hides the Forms toolbar. [**WHERE IS IT?**]

He goes back to the LI button, inserts a fourth <LI> tag but immediately erases it. [**OOPS!**]

He then presses the neighboring Bot button, which inserts an application-specific tag, and again immediately deletes it. [**OOPS!**]

He hesitates a little bit, moves the window, scrolls the document up and down, *and declares that he has completed the task, without even attempting to preview the generated page.* [**LOOKS FINE TO ME.**]

User’s interaction with SpiderPad:

The application starts with a blank document. The user selects the HTML tag from the Structure toolbar menu, and then the BODY tag. The application inserts both tags in the same line. The user moves the cursor in between the tags and presses the Enter key to move the BODY tag one line down. The user opens the Lists, Heading and Structure menu, looking for the file header. From the Structure menu, the user selects the TITLE tag. The application inserts both opening and closing tags. The user types in ‘‘Teste’’ and places the cursor after the BODY tag. He opens the Font, Heading, List and Structure toolbar menus. Then the user passes the mouse over several toolbar buttons, and opens the Insert pull-down menu. He hovers over the Modify pull-down menu, but immediately returns to the Insert menu and selects the ‘‘Body Tag’’ item. The application presents the ‘‘Modify Body’’ dialog box. The user changes the background color to white. The application inserts the BGCOLOR attribute within the BODY tag. The user types in the closing </BODY> tag. He looks for some way of inserting a list in the Tags menu, and chooses the Unordered list menu item (which is the wrong type of list). The application inserts both opening and closing UL tags. He types in the two items, goes back to the Tags menu, hesitates for a very little while and then chooses the Ordered list menu item. The application inserts both opening and closing OL tags. Realizing the first choice was wrong, the user erases the opening UL tag, and cuts the two items, pasting them inside the newly inserted OL tags. Next, he erases the extra UL closing tag. He moves the cursor after the first item, selects the ‘‘List item’’ menu item in the Tags menu and types the first item of the second list. He repeats these actions for the two remaining items. He cuts the closing OL tags and

> pastes it between the two lists. Finally, he types in a closing </LI> tag after the last item of the second list, and declares that he has completed the task, without even attempting to preview the generated page.

In the following we repeat the above narration, indicating in boldface the tags that the evaluators assigned to segments of the movie, formatted in underlined text.

> The application starts with a blank document. The user selects the HTML tag from the Structure toolbar menu, and then the BODY tag. The application inserts both tags in the same line. The user moves the cursor in between the tags and presses the Enter key to move the BODY tag one line down.
>
> *The user opens the Lists, Heading and Structure menu, looking for the file header.* [**WHERE IS IT?**]
>
> From the Structure menu, the user selects the TITLE tag. The application inserts both opening and closing tags. The user types in "Teste" and places the cursor after the BODY tag.
>
> *He opens the Font, Heading, List and Structure toolbar menus. Then the user passes the mouse over several toolbar buttons, and opens the Insert pull-down menu. He hovers over the Modify pull-down menu*, but immediately returns to the Insert menu and selects the "Body Tag" item. [**WHERE IS IT?**]
>
> The application presents the "Modify Body" dialog box. The user changes the background color to white. The application inserts the BGCOLOR attribute within the BODY tag.
>
> *The user types in the closing* tag. [**I CAN DO OTHERWISE.**]
>
> *He looks for some way of inserting a list in the Tags menu*, and chooses the Unordered list menu item (which is the wrong type of list). [**WHERE IS IT?**]
>
> The application inserts both opening and closing UL tags. He types in the two items, goes back to the Tags menu, hesitates for a very little while and then chooses the Ordered list menu item. The application inserts both opening and closing OL tags. Realizing the first choice was wrong, the user erases the opening UL tag, and cuts the two items, pasting them inside the newly inserted OL tags. Next, he erases the extra UL closing tag. He moves the cursor after the first item, selects the "List item" menu item in the Tags menu and types the first item of the second list. He repeats these actions for the two remaining items. He cuts the closing OL tags and pastes it between the two lists.
>
> Finally, he types in a closing <LI> tag after the last item of the second list, and *declares that he has completed the task, without even attempting to preview the generated page.* [**LOOKS FINE TO ME.**]

Pre-Test & Post-Test Interviews

The pre-test interview asked participants about:

1. What kinds of tools did they use to create HTML pages?
2. How did they do it?
3. What tools did they use to modify existing pages?

4. How many HTML pages had they already created?
5. What level of expertise in HTML did they think they had?
6. What was their favorite *text* editor?
7. What operating system did they use (for web publishing and related activities)?

In addition to disambiguating portions of the observed interaction, in the post-test interview the evaluator asked participants about:

1. What were the perceived differences between Arachnophilia and SpiderPad?
2. Which of the two editors did they prefer and why?
3. What kind of user did they think the HTML editors were designed for? Why?
4. Which frequent tasks did each editor support best? Why?
5. Did they think *they* were targeted users of Arachnophilia and/or SpiderPad? Why?
6. In which of the two editors is it easier to *create* a table? Why?
7. In which of the two editors is it easier to *modify* a table? Why?

Observer's Annotations

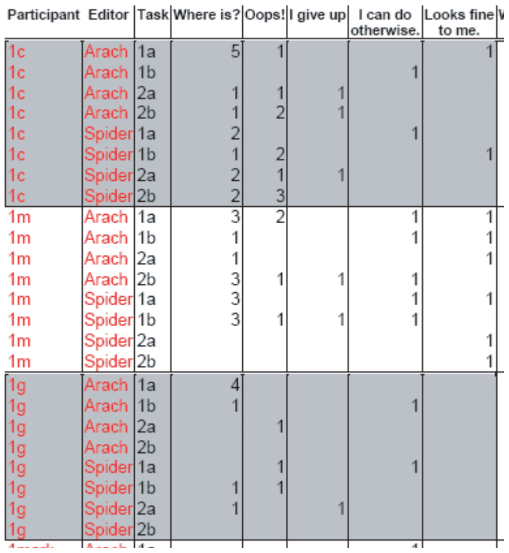
Here are two examples of useful annotations made by the observer during the interactive sessions.

> "*Participant X is typing the HTML tags directly; only uses the editor's tag tools if he is not sure of which tag he should use (+ attributes, etc).*" [Notice the observed reason for the participant's not using some of the editor's tag tools. This reason is inferred from various observed signs: the participant's typing speed, body posture, facial expression, etc.]
>
> *''Because Participant Y has used Arachnophilia first, he is confused with SpiderPad's blank page—is there additional code hiding somewhere? See how he opens the edited page code in Notepad (!) in order to see if there is something else there.''* [Notice the observer's instant interpretation of what is going on in this participant's mind. This interpretation is supported by the whole context of the text, in which the observer is fully immersed.]

Tabulation and Interpretation

Here are examples of tabulations and interpretations produced during CEM.

In Figure 2 we see a portion of the tabulations of *tags x participant x task*. Notice how participants differ in their communicative experience. Whereas some have difficulty in finding the appropriate way to express themselves (''Where is it?) in both editors (see participant 1m), others seem to have no difficulty with one of them (see participant 1g with SpiderPad). Also, in some cases, some kinds of communicative breakdown seem to decrease over time (see *''Oops!''* for participant 1m's progressive tasks with Arachnophilia), whereas others don't (see *''Oops!''* for participant 1c's progressive tasks with SpiderPad).



Participant	Editor	Task	Where is?	Oops!	I give up	I can do otherwise.	Looks fine to me.
1c	Arach	1a	5	1			1
1c	Arach	1b				1	
1c	Arach	2a	1	1	1		
1c	Arach	2b	1	2	1		
1c	Spider	1a	2			1	
1c	Spider	1b	1	2			1
1c	Spider	2a	2	1	1		
1c	Spider	2b	2	3			
1m	Arach	1a	3	2		1	1
1m	Arach	1b	1			1	1
1m	Arach	2a	1				1
1m	Arach	2b	3	1	1	1	
1m	Spider	1a	3			1	1
1m	Spider	1b	3	1	1	1	
1m	Spider	2a					1
1m	Spider	2b					1
1g	Arach	1a	4				
1g	Arach	1b	1			1	
1g	Arach	2a		1			
1g	Arach	2b					
1g	Spider	1a		1		1	
1g	Spider	1b	1	1			
1g	Spider	2a	1		1		
1g	Spider	2b					

Figure 2 *A snapshot of a complete tag/participant/task tabulation*

The interpretation of tagged movies, tabulations, observations and interviews, allowed the evaluator to draw interesting conclusions about the editors. There are basically two main factors to explore in comparing the editors:

- First, both Arachnophilia and SpiderPad caused considerable problems of navigation for the participants (a high frequency of *''Where is it?''*). It was also difficult, in both editors, to assign meanings to many interface symbols. But the frequency of meaning-assigning problems with SpiderPad was higher than with Arachnophilia. This is a curious result, given that most participants explicitly said, in the post-test interview, that SpiderPad was *easier* than Arachnophilia. However, the frequency of communicative breakdowns directly associated to (sub)task failures (*''I give up*'' and *''Looks fine to me''*) was slightly better in SpiderPad (31 hits) than in Arachnophilia (36 hits).
- Second, Arachnophilia was somewhat more *conversational* than SpiderPad, in that it had a smaller number of hits (97) than SpiderPad (108) for tags like *''Where is it?'', ''Oops!'', ''What's this?'', ''What happened?''* and *''Why doesn't it?''*. This observation is in line with the kind of discourse we find in each editor's help contents. SpiderPad's help is terse and impersonal (e.g. the designer gives the following instruction for adding a row/column to a table: ''To add a row or column, select a cell and click the appropriate button. Rows are added above the row of the selected cell, and columns are added to the left of the selected cell.''). Arachnophilia's help style, however, is quite the opposite. The designer directly addresses the users and explicitly stands as the first person in

discourse (we even know the designer's name), as evidenced by phrases like *"I can't know what your background is or how much you know about computers, so you may choose. . ."* (help content for topic *How to make your own page*). Moreover, in Arachnophilia help is organized in a tutorial way, whereas in SpiderPad it is organized in a functional way.

The Semiotic Profile of Arachnophilia & SpiderPad

The answers to the 5 questions for the semiotic profiling stage allow us to see clearly some of the main differences between Arachnophilia and SpiderPad.

1. *Who do the designers think are the users of the product of their design? (i.e. Who are the apparent receivers of the designers' communication?)*
   CEM suggests that Arachnophilia's designer is talking to HTML learners—people who don't know HTML, but are eager to learn. SpiderPad's designer, however, is talking to HTML coders—people who know enough about HTML, and would be glad to have a tool to accelerate their coding.
2. *What do they think are these users' wants and needs? (i.e. how is communication tailored to privilege certain wants and needs, and not others?)*
   Arachnophilia's designer provides tips and explanations throughout interaction, adopting a *verbose* style that is typically used for novices. SpiderPad's designer provides functions that build larger portions of HTML coding.
3. *Which do they think are these users' preferences with respect to their wants and needs, and why? (i.e. how and why is the* users' *communication with the system facilitated in certain contexts, and not others? Are there communicative choices available?)*
   Through long dialogs, and extensive use of comments in the generated HTML code, Arachnophilia's designer places a high value on the user's ability (and desire) to *learn by doing*. SpiderPad's design puts the user in control, and behaves in a *reactive* way, assuming that *the user knows what he is doing*.
4. *What system have they* consequently *designed for these users, and how can or should they use it? (i.e. how consistently do the design principles match the designers' communication about the users they expect to be talking to?)*
   Arachnophilia is a *tutorial* tool for HTML coding, whereas SpiderPad is a *toolbox* itself. Both applications provide powerful customization and extension facilities (i.e. template creation, macro-programming, etc.). However, the gap in Arachnophilia is much larger than in SpiderPad. The tutorial tone in Arachnophilia breaks down when advanced programming features are introduced. In SpiderPad, however, advanced programming is totally in line with the "expert user profile" that the designer addresses in all communications (especially through online help content).

5. *What is the designers'* global design vision*? (i.e. How is the technology made available through the system expected to be received, and to transform the users' lives in a positive way?)*
   Arachnophilia is clearly expected to offer users a positive learning experience, whereas SpiderPad is designed to facilitate the coding effort. The challenge for Arachnophilia is to *grow with the learner*. The design is so densely populated with signs whose communicative intent is to *teach*, that advanced users may have a negative reaction to the technology. In fact, some of the participants in this evaluation explicitly said that they didn't like Arachnophilia because it treated them as *beginners*. Of course, conversely, SpiderPad doesn't help beginners at all. This is an important result of CEM, since a function-by-function comparison between both editors shows that they have a lot in common, and that Arachnophilia can even support some very advanced coding tasks that SpiderPad cannot. But because the communication is so clearly addressed to different interlocutors, the functionality may never get to be used.

Bibliography

Semiotic Engineering

- **de SOUZA, C. S.** (2005) The semiotic engineering of human-computer interaction. Cambridge, MA. The MIT Press

Introduction to CEM

- **PRATES, R. O.; de SOUZA, C. S; BARBOSA, S. D. J.** (2000) "A Method for Evaluating the Communicability of User Interfaces". ACM interactions, Jan-Feb 2000. pp 31–38.

Observing the Users' Learning Curve

- **PRATES, R. O.; BARBOSA, S. D. J.; de SOUZA, C. S.** "A Case Study for Evaluating Interface Design through Communicability". *Proceedings of the ACM Conference on Designing Interactive Systems, DIS'2000.* Agosto de 2000. pp. 308–316.

Operational, Tactical and Strategic Communicability Breakdowns (JBCS, 2000)

- **de SOUZA, C. S.; PRATES, R. O.; CAREY, T.** "Missing and declining affordances: Are these appropriate concepts?". *Journal of the Brazilian Computer Society,* SBC. Porto Alegre, RS, v. 7, n. 1, p. 26–34, 2000.

Communicability Evaluation with different technologies (Prates, 2001; Prates, 2004)

- **de SOUZA, C. S.; PRATES, R. O.; de ASSIS, P. S.** (2001) “Categorizing communicability evaluation breakdowns in groupware applications”. *2nd South African Conference on Human-Computer Interaction (CHI-SA 2001)*,10–12 of September, 2001.
- **PRATES, R. O.; LEITÃO, C. F.; FIGUEIREDO, R. M. V.** “Desafios de Avaliação de Interfaces de Ambientes Educacionais—Um Estudo de Caso”. In: *VI Simpósio sobre Fatores Humanos em Sistemas Computacionais, IHC’2004*. 2004, Curitiba. Anais do VI Simpósio sobre Fatores Humanos em Sistemas Computacionais. Porto Alegre : Sociedade Brasileira de Computação, 2004. p. 185–188.

SERG Website: `http://www.serg.inf.puc-rio.br`