3 WNH: Web Navigation Helper

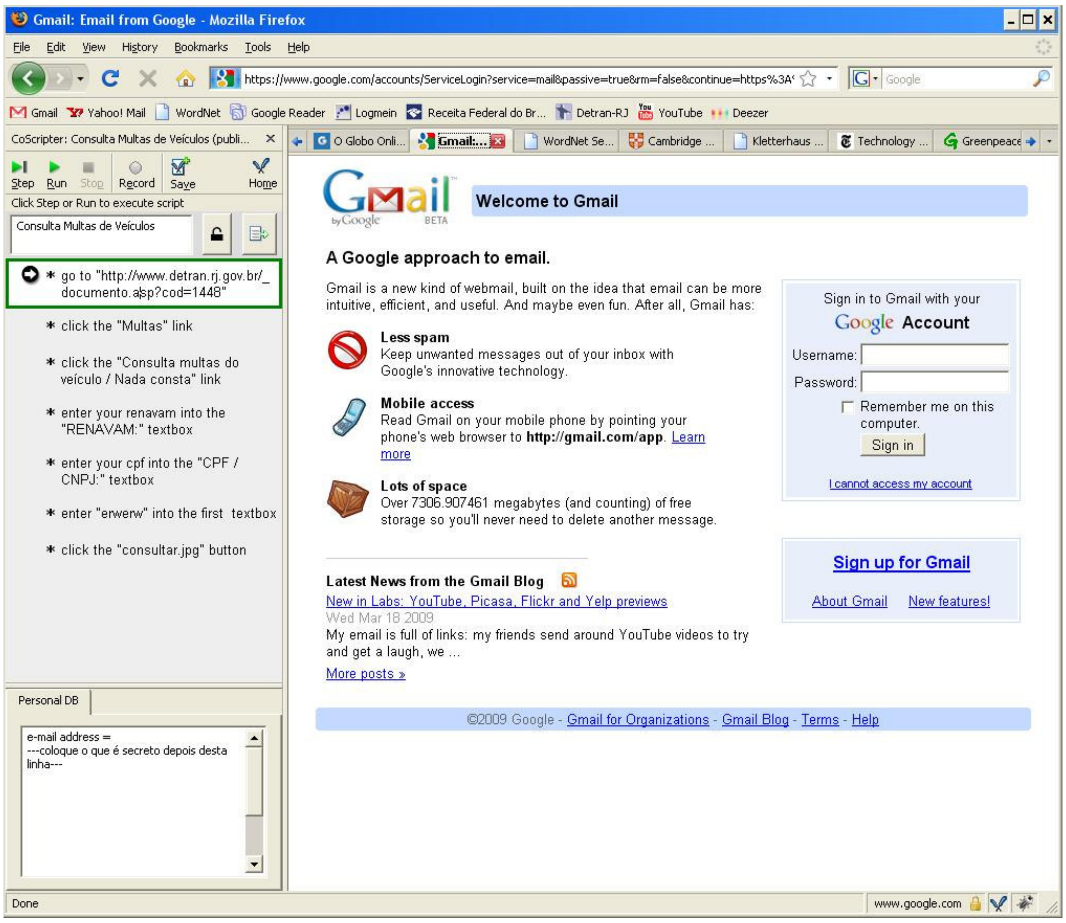
This chapter presents a detailed description about Web Navigation Helper (WNH), the web application developed in this research. As mentioned in section 1, WNH is based on a tool called CoScripter. In order to better understand WNH, section 3.1 introduces the CoScripter tool and its purposes. In section 3.2, a detailed description of WNH is presented, followed by its architecture in section 3.3. Section 3.4 presents an illustration of WNH and section 3.5 alerts to the other side of the coin: where and how it could go wrong, and what can be done to overcome or recover from these potential failures.

3.1. CoScripter

CoScripter is a macro-recorder for the web, developed at IBM Research in Almaden (CoScripter, 2008). "*CoScripter is a tool that brings together known ideas in a novel combination: 1) it allows end-user automation of procedures through recording and scripting, and 2) it stores scripts on a shared central wiki*" (CoScripter, 2008).

CoScripter enables users to record interactive tasks carried out with a browser and to play them back later. This is a nice-to-have functionality, especially when dealing with processes that are frequently performed or are too long to be manually repeated. The set of these recorded actions is called script. Once recorded, scripts can be easily reproduced. Filling in forms, clicking on buttons, navigating to specific URLs are some of the actions that can be easily automated through CoScripter, currently implemented as an extension for the Mozilla Firefox web browser and written in JavaScript and XUL. Access to CoScripter is provided upon registration, and every member has access to CoScripter's forum and wiki, and is able to share with and see shared scripts from other members.

The CoScripter interface is a bar located on the left side of the browser, as shown in Figure 2.


Gmail: Email from Google - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=https%3A
Gmail Yahoo! Mail WordNet Google Reader Logmein Receita Federal do Br... Detran-RJ YouTube Deezer
CoScripter: Consulta Multas de Veículos (publi...
O Globo Onli... Gmail:... WordNet Se... Cambridge ... Kletterhaus ... Technology ... Greenpeace
Step Run Stop Record Save Home
Click Step or Run to execute script
Consulta Multas de Veículos
Personal DB
Done
www.google.com

* go to "http://www.detran.rj.gov.br/_documento.asp?cod=1448"
* click the "Multas" link
* click the "Consulta multas do veículo / Nada consta" link
* enter your renavam into the "RENAVAM:" textbox
* enter your cpf into the "CPF / CNPJ:" textbox
* enter "erwerw" into the first textbox
* click the "consultar.jpg" button

e-mail address =
---coloque o que é secreto depois desta linha---

Welcome to Gmail

A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

Less spam
Keep unwanted messages out of your inbox with Google's innovative technology.

Mobile access
Read Gmail on your mobile phone by pointing your phone's web browser to **http://gmail.com/app**. Learn more

Lots of space
Over 7306.907461 megabytes (and counting) of free storage so you'll never need to delete another message.

Sign in to Gmail with your Google Account
Username:
Password:
Remember me on this computer.
Sign in
I cannot access my account

Sign up for Gmail
About Gmail New features!

Latest News from the Gmail Blog
New in Labs: YouTube, Picasa, Flickr and Yelp previews
Wed Mar 18 2009
My email is full of links: my friends send around YouTube videos to try and get a laugh, we ...
More posts »

©2009 Google - Gmail for Organizations - Gmail Blog - Terms - Help

Figure 2: CoScripter

Every script is associated to the URLs in it. So, for example, if a script automates the login process of a user to Gmail, this script is immediately associated to www.gmail.com.

CoScripter has an important collaborative aspect that turns it into a powerful tool, and is of major significance to this work. Every new script created is designated by its owner as private or public. Public scripts can be seen and used by all members of the CoScripter community, while private ones are accessed and viewed only by their creators. When visiting a site, CoScripter exhibits to users a list containing all public scripts related to that site. Should they wish to run a script, they click on it to load the script in CoScripter's sidebar, and click on "Run" to execute it. Users can benefit from shared scripts since they don't need to create their own script to perform these actions. They may use somebody else's scripts.

Suppose a user visits a web page that contains shared CoScripter scripts and, by browsing this list, she identifies, amongst all shared scripts related to that page, one script that performs the exact set of actions she originally intended to perform in that web page. Instead of running the actions one by one, the user has the option to click and run the script. By doing so, the actions will be automatically ran. The benefits of selecting, loading and running a script, instead of manually executing the set of actions contained in the script, are manifold:

1) It should be less time-consuming;

2) Users that were not sure about how to manually perform the tasks find support on this automatic way;

3) Users with difficulties in navigating the web, as for instance the visually-impaired and the functionally-illiterate, don't need to browse the site content in search for the requested information;

Every promising attempt of guiding the user to the resource she is looking for, preventing her from getting lost in the midst of the site contents, should be pursued further. With that said, however, the tool still presents few challenges to make it happen. Two of the challenges can be said to impact all users:

1) How can users tell which script, from the list associated to the page they are visiting, does the task they wish to do?

2) How do users know which information she should provide (insert into the personal database) in order to execute the steps?

Another challenge concerns users with special needs. How would users less acquainted with computers and programming manage to perform these tasks? What about visually impaired and functionally illiterate users, could they take any profit of it? Or would they face way too many difficulties that it would not be worth it?

As shown here, using CoScripter might represent to some users an effort not easy to overcome. This is where WNH comes to play, as described in next section.

3.2. Detailed Description

The challenges raised in last section suggest the fact that interacting with CoScripter might require basic familiarity with computers and Internet and even some programming skills. Nevertheless, the intention was to bring CoScripter and some of its functionalities to novice users, as it might become an efficient way of accessing resources in the web. It was found however essential to review the way CoScripter is presented to end-users: certainly not all of its features would be of use to the targeted public. Moreover, these end-users would probably require a more assistive way of executing the scripts than more expert users. These considerations (among others that will be explained later in this chapter) were the driving force to build for the targeted web-novice users a whole supportive solution of sharing and executing automated processes. The Web Navigation Helper (WNH) was then conceived.

The main purposes of WNH are threefold:

1) Allow end-users to easily locate processes (scripts) that will bring them to the desired resource;

2) Allow end-users to automatically execute those processes (which is already contemplated by CoScripter);

3) Assist end-user during the processes executions;

WNH, nevertheless, includes more than the topics just mentioned. As will be seen later on this chapter, it also comprises few features to assist those interested in creating and maintaining shared scripts to be used by the end-users. In other words, it is also designed to help those wanting to help. Since WNH is a solution to be used by three distinct groups (blind, functionally illiterate and support users), it was important to separate it into three different tools, so users would not be erroneously distracted with features designed to the other group. From now on in the text, the terms **WNH-see**, **WNH-read** and **WNH-support** will be used when referring to the solution for blind, functionally illiterate and support users, respectively. It is important to mention that these three tools are generated by the same original code, with different compilation flags turned on/off during the code process build, depending on which public the application is targeted to.

Since the universe of sites and scripts is enormous, with diverse characteristics and demanding different approaches of solutions, it was chosen in this study to restrict the sites' domain to Brazilian government only. It was expected with this to collect enough information and learn from this universe so, in future works, the solution can be easily extended to more sites. Also, to be able to work with relevant scripts, a group of three volunteers created a database of useful scripts related to Brazilian govern sites. Few Brazilian government sites that gathered different, useful and common tasks were chosen, and for each one of them, tasks that attended to at least one of the requirements listed below were selected:

1) Useful to users (such as checking the income tax refund);

2) Common to users (tasks often executed by users);

3) Have a *dialog* with the user, or in other words, tasks that require some data input from the user in order to be executed (for instance, tasks that ask for the ID card number);

By doing so, a number of useful scripts from distinct sites were created. These scripts served not only as the starting point for the experiments, but also as the repository with which the WNH was and is being tested and developed.

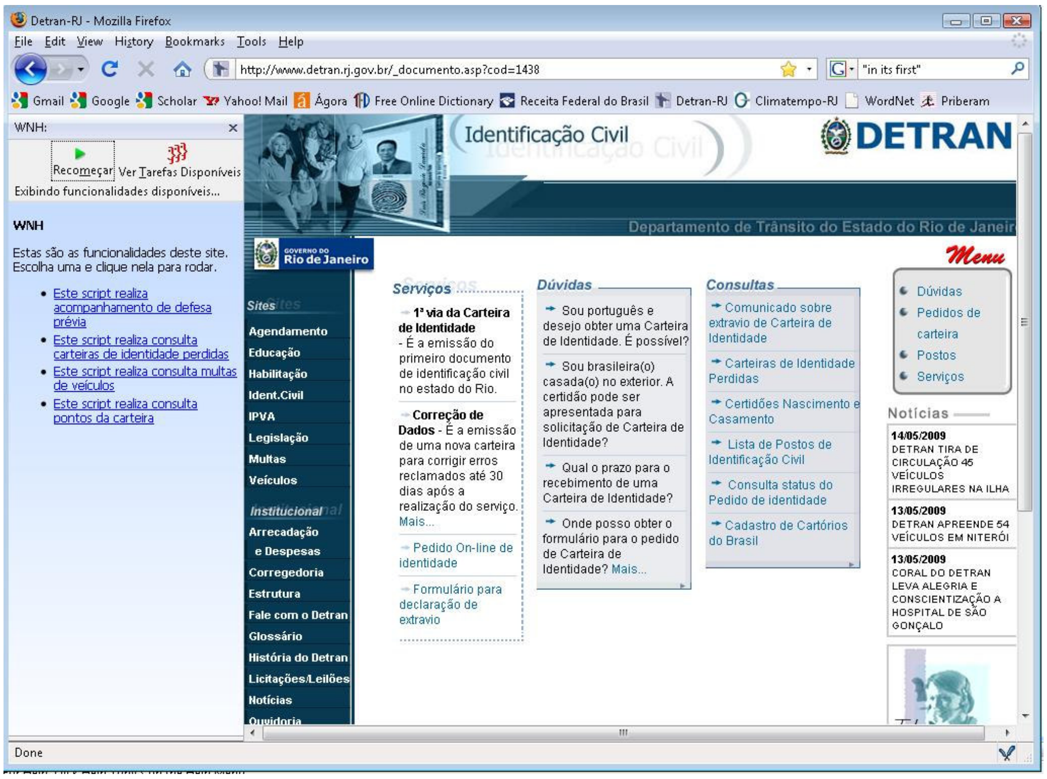
3.2.1. WNH-see and WNH-read

All three versions WNH-see, WNH-read and WNH-support are built on top of CoScripter, as all should be able to reproduce automated tasks, find out shared tasks, execute them, and so on. On the backend of the three tools runs the same CoScripter machine[7]. WNH-see and WNH-read, however, are different from WNH-support since they omit a number of elements from the original CoScripter that are no longer useful for end-users, and brings to end-users only some of the CoScripter original features (basically, only those needed to run the scripts).

Both WNH-see and WNH-read can be seen as a layer wrapping CoScripter, including features and filtering others that are unnecessary, so as not to disturb the targeted end-users with interface elements they won't anyway use

[7] CoScripter machine can be understood as the capabilities of creating, sharing, exhibiting and running automated processes

(and mistakenly distract them from their real activity). Actually, only the essential functionalities were left. Most of the original interface elements (e.g. the ‘Record’ and ‘Save’ buttons, as shown in Figure 2) were suppressed, and the end-user is left on the initial page with two buttons only. One button, called “Available Tasks”, once clicked displays a list of shared scripts (i.e., public scripts) referent to the web site currently displayed on the main browser window (as explained in previous section). The button acts also much like the “Home” button in a browser: it is always visible and accessible and, when clicked, resets the process by displaying the same initial list (as can be seen in Figure 3). The second button, called “Restart”, allows for restarting any on-going process execution.



Exibindo funcionalidades disponíveis...

WNH

Estas são as funcionalidades deste site. Escolha uma e clique nela para rodar.

- Este script realiza acompanhamento de defesa prévia
- Este script realiza consulta carteiras de identidade perdidas
- Este script realiza consulta multas de veículos
- Este script realiza consulta pontos da carteira

Serviços

1ª via da Carteira de Identidade - É a emissão do primeiro documento de identificação civil no estado do Rio.

Correção de Dados - É a emissão de uma nova carteira para corrigir erros reclamados até 30 dias após a realização do serviço. Mais...

Pedido On-line de identidade

Formulário para declaração de extravio

Dúvidas

Sou português e desejo obter uma Carteira de Identidade. É possível?

Sou brasileira(o) casada(o) no exterior. A certidão pode ser apresentada para solicitação de Carteira de Identidade?

Qual o prazo para o recebimento de uma Carteira de Identidade?

Onde posso obter o formulário para o pedido de Carteira de Identidade? Mais...

Consultas

Comunicado sobre extravio de Carteira de Identidade

Carteiras de Identidade Perdidas

Certidões Nascimento e Casamento

Lista de Postos de Identificação Civil

Consulta status do Pedido de identidade

Cadastro de Cartórios do Brasil

Notícias

14/05/2009 DETRAN TIRA DE CIRCULAÇÃO 45 VEÍCULOS IRREGULARES NA ILHA

13/05/2009 DETRAN APREENDE 54 VEÍCULOS EM NITERÓI

13/05/2009 CORAL DO DETRAN LEVA ALEGRIA E CONSCIENTIZAÇÃO A HOSPITAL DE SÃO GONÇALO

Figure 3: Web Navigation Helper

More than the changes in the interface level, the way WNH-see and WNH-read interact with end-users was also reviewed. As mentioned before, blind and functionally illiterate users might present difficulties in interacting with the tool. Therefore WNH-see and WNH-read were conceived to serve as an interceptor during CoScripter execution so as to help users in each and every step that can be considered to be a point of possible errors, allowing them, then, to accomplish the task. For instance, opposed to the way CoScripter requests personal information in order to run the scripts (it does that by presupposing the user has already filled in a variable in his main Personal Database, so it can be retrieved when running the script), both WNH-see and WNH-read do not store variables values in any database. All information is requested on-the-fly, through a pop-up window[8] (and discarded after the script completion). The original Personal Database can be seen as a place where users could store values of variables frequently used. But in the case of WNH this is not really appropriate. On the contrary: the whole idea of variables and values are a notion reserved to programmers and/or computers expert users, and would be an obstacle for the targeted users. Asking for the needed information during the task execution presented itself as a more natural and familiar way than doing that through the Personal Database.

Another functional innovation implemented in WNH, which plays a major role in the solution, is the communication channel created from support-user to end-user. Through this one-way channel, support-users are able to orient and instruct end-users during their script execution. Every problem and difficulty faced by end-users during the process could be guided by support-users. The encountered way to implement this is through the comments. Therefore, in WNH-see/WNH-read, every comment in the script is presented to the end-user in a pop-up message window. Whenever a comment is recognized during the script execution, a pop-up confirmation window is displayed to users with the comment contents. This creates an important and valuable channel of communication between support and end-users.

Acting as an interceptor to assist users during script execution turns WNH into a powerful tool, since it creates hooks during the process in which several new functionalities could be attached. In other words, the fact that CoScripter process execution can be intercepted during its steps flow permits incorporating several actions intended to assist users. One could think, as an illustration of possibilities for the future, of an online live support chatting environment to guide end-users during the script execution: whenever the user is uncertain of what to do next, she could ask for assistance from the support team.

The way WNH-see/WNH-read accomplishes these interceptions is by running over the script steps, interpreting every individual command of the steps,

[8] Unless contrarily indicated, in order to save space in the text, every mention of a window pop-up to provide or request information to/from end-user should be understood that WNH-see for blind users performs the same parallel interaction through the use of screen readers, by reading out-loud the information.

collecting information about the required input data, and according to its internal rules, deciding when the interception is needed. WNH-see/WNH-read is also able to retrieve the contents and extract relevant information from the web page itself. That increases even more the range of assistive actions that can be performed. Clearly, WNH infrastructure presents itself as quite valuable to users: it provides the basics for creating a dedicated and encompassing system for blind and functionally illiterate internet, (but not only them), end-users.

3.2.2. Differences between WNH-see and WNH-read

Visually, WNH-see almost does not differ from WNH-read. Every change and modification to WNH-read was carried out to WNH-see too, mostly because WNH-see users would not notice these differences anyway (because of their blindness). This scenario probably won't be the same when WNH is extended to other visually impaired users. One interface particularity, though, is present on WNH-see but not on WNH-read, and it will be described immediately after the main difference between these two tools is presented in the next paragraph.

The differences between WNH-see and WNH-read rely mostly on how they interact with the end-user. As will be further explored in sections 4 and 5, during the WNH-see first experiment it was noticed that the user had an issue in locating himself during the automated process execution. He was not informed of what was taking place in every moment, what was already done, and what was left to be done. With that in mind, it was understood that the blind users need to be notified at every step about his current position in the whole process. The encountered solution was the creation of a pop-up message that informs, at every concluded step, the accomplishment of x steps out of a total of y steps. This information is not displayed to WNH-read users, since they are visually notified of the process progress during CoScripter machine execution.

Getting back to the interface particularity mentioned above, whenever this pop-up message is displayed during the steps execution in WNH-see, a Help and a Restart buttons are displayed to the blind users as well, but not to the functionally illiterate users. The Help button was an attempt to reduce the sense of loss of orientation manifested by the first blind experiment participant, which could also be shared by the others during the experiments; the Restart button was a way of providing errors recovery mechanisms to the user. Since the Restart functionality is already available in WNH main interface and can be easily accessed by sighted users, there is no need of displaying the Restart button at every step to WNH-read users in a separate message. As for the Help button, it was not considered to be of major importance in current version of WNH-read, and therefore it was omitted from it. This explains why these two buttons are present in WNH-see but not in WNH-read. Future versions of WNH-read should consider, though, making use of the Help as it is an important channel of conversation between designers and end-users.

3.2.3. WNH-support

As mentioned above, WNH-see/WNH-read and WNH-support are two faces of the same tool, intended to be used by end-users and support-users, respectively. In this work the notion of a *support* group is used, meaning a group composed of volunteers in charge of the following tasks:

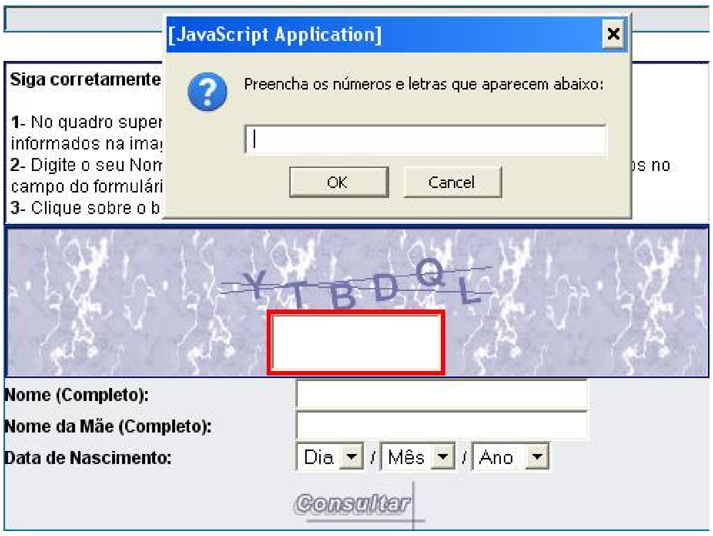
1) Create useful scripts, primarily in Brazilian government sites, to be used by WNH-see/WNH-read users;

2) Maintain these scripts, which means adapting them to eventual changes undergone by sites;

When creating new scripts, and in order to make the most out of WNH, volunteers should fully comprehend important points. The first of them is how WNH works and its features. Also, they should have a good understanding of the web site in which the script is being created, and the possible ways of accomplishing the task. Finally, they should have a very good understanding of the end-users' characteristics, their most common difficulties, and plan the script steps accordingly to that. For instance, many of the government sites (and not only them) make use of *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA) mechanism used to prevent automated navigation, and which is a well-known issue in the accessibility field. Volunteer users, when creating scripts and facing a captcha step, should try to prevent possible users falls by providing the necessary information. A suggested way of providing this information could be as follows:

"enter your 'os números e letras que aparecem abaixo' into the first textbox"

That command, when intercepted by WNH (and then by the CoScripter machine), will pause the execution of the process for user data input. WNH then pops-up a window requiring such data from user, as shown in Figure 4.


[JavaScript Application]

Preencha os números e letras que aparecem abaixo:

Siga corretamente

1- No quadro super informados na ima

2- Digite o seu Nom campo do formulári

3- Clique sobre o b

Nome (Completo):

Nome da Mãe (Completo):

Data de Nascimento:

Figure 4: *captcha* when running WNH

Whatever is enclosed in single-quotes in this command (e.g. 'os números e letras que aparecem abaixo') is displayed to end-users, and constitutes the door to guide users. This text will be used by volunteer users to better explain to ebd users what needs to be filled in the box.Therefore, support users should carefully prepare this text, as in the example above, in which users are told to “fill in the box with the numbers and characters displayed below”.

Besides the task of creating scripts, it is up to the volunteers to keep these scripts *up and running* as well. But how are volunteers supposed to maintain the scripts functional and active? This is especially important when talking about web sites, a vehicle liable to sudden changes. Not only there are no guarantees the site seen today will be the same tomorrow, but also the changes are fairly common. A script might be no longer functional if any change was made to the site. In order to find out if scripts are still suitable to their sites, volunteers would need to either check if there was any change to this site, or manually run the scripts periodically, adapting the script if needed.

WNH-support assists volunteers by providing a tool that automatically performs this “finding-out changes” task. The tool should be run periodically, and for a list of sites and scripts, it executes each script, simulating any required user data input, and checking if the script reached or not the end. If for any reason it stopped before the end, it raises a suspect flag, and should be closely analyzed by a volunteer. If the script successfully reached the end, then the script does match its site.

The mission undertaken by the volunteers is not an easy one. It is up to them to create scripts that not only are *runnable* and do execute the desired task, but that are user friendly and facilitate the end-users' understanding. Volunteers must take into account all possible and eventual difficulties the users might have in each and every step of the process, such as the audio captcha, for instance. In this case, it is a good practice to add, before the correspondent command, a comment with some sayings such as "next you will hear numbers that need to be typed in", which will be shown to the user as a notification. Or, in the functionally-illiterate case, the best practices indicate that a comment also needs to be added before the point in which the user is supposed to interact directly with the web page. This comment should inform the user how to trigger the execution back.

The examples presented above are just some of what was learnt from the first two experiments. As the number of scripts and tasks start to grow, there will probably emerge other essential particularities, not yet foreseen, which must not be put aside. As shown here, the role of the volunteers is of greatest importance for the success of this project.

Future versions of WNH-support should consider the creation of a markup language to be used by the volunteers in order to construct more efficient dialogues with the end-users.

Sections 3.2.1 and 3.2.3 present how WNH looks like today, after few months of natural evolution. The next section will present a brief explanation of this evolution, how it all started, since it is important to understand the considerations taken throughout the process and which ways seemed more appropriate to pursue. This will serve as a basis for further work as well.

3.2.4. History of the project

In its early stage, still aiming at improving the web navigation through the use of scripts, the means by which this project tried to accomplish this was different from the current one. Then, the focus was to help users in choosing the CoScripter script associated to a web site of interest that performed the task they needed. Although short scripts (e. g. with less than 5 instructions) are likely to be easy to understand, longer scripts may not be so easy to interpret. As such, the original idea of this project was to create a tool that would help users identify their scripts of interest. Moreover, the tool would help users during the script execution, by giving them hints and tips concerning what information they should provide. Aiming at these two goals, the concrete purpose of the first versions of WNH was twofold: (a) it would provide useful descriptions of what scripts do (especially longer scripts); and (b) it would inform users about the kinds of input that are required for the script to execute. Two possibilities for accomplishing the '(a)' purpose were raised, named plan A and B. In plan A the descriptions were to be generated automatically by what was called the Pragmatic Script Interpreter (PSI). The idea of PSI was to parse the contents of every individual shared script related to the site, extract relevant information from it and generate descriptions of their purposes in natural language, so as to be fully comprehended by users. Whilst plan A represented the automatic way, plan B was the opposite extreme. In the plan B scenario, the descriptions, as well as the scripts themselves, would be generated by a group of volunteers engaged in the project.

However, during the development of the project, and due to time considerations, plan A was put aside for future work and the research headed mainly toward plan B. Because the descriptions were now manually provided by volunteers, there was no more point in automatically interpreting scripts. In other words, the Pragmatic Script Interpreter's functionality would be temporarily suspended. The 'PSI' name was hence discarded and a more proper name, 'Web Navigation Helper' (WNH), came about.

When WNH emerged, it was initially idealized and prototyped mainly as one single solution to both blind and functionally illiterate users, with minor differences between them. For instance, WNH for the blind would make use of screen readers during their navigation, while the others could use screen readers or not. However, as it was learnt from the first experiments carried out during the project, functionally illiterate users proved to be a population with significantly different needs from those of blind users. In order to best attend to the requirements of both populations, a decision to divide into two the original single solution was made, and the differences between the two are described in previous section 3.2.2. The result of this decision was that, in its general conception, WNH, as its names suggests, is a helper for web navigation for both blind and functionally illiterate users, but when closely analyzed, the differences that address each of the types of users are vast, demanding the maintenance of two parallel versions.

3.3. Architecture

CoScripter (CS) is essentially a web system. It was developed as a Mozilla Firefox extension, and therefore obeys its add-ons development guidelines (Mozilla Firefox Co., 2009). Every Firefox extension is built using four technologies, namely: XUL, JavaScript, CSS and XPCOM (Mozilla Firefox Co., 2009). It is not in the scope of this work to enter the details of Firefox add-ons implementation. Rather, it should be enough to give a brief overview of each technology in order to better understand how CS is built and where the changes were applied.

1) XUL (XML User Interface Language) – an XML language for describing user interfaces. The elements in the XUL documents define the layout of the application user interface.

2) JavaScript – a scripting language primarily used for writing functions to be embedded in or included from HTML pages. It enables programmatic access to objects within other applications.

3) CSS (Cascade Style Sheet) - a language used to describe the appearance (that is, the look and formatting) of a document which was written in markup language (HTML, XML, etc).

4) XPCOM (Cross Platform Component Object Model) – this is what enables the integration of external libraries with XUL applications.

The logic and appearance of the developed Firefox extension based on the technologies listed above should attend a pre-defined directory structure. This organization should then be compressed in a ZIP file and renamed to '.xpi' extension in order to be distributed and installed in the browser.

It is important to alert that CoScripter is a Client-Server web system, and as such it has a counterpart on the server-side. The client-side of CoScripter is the Firefox add-on, and was therefore developed mainly in JavaScript and XUL languages, with some components in CSS and XPCOM. Because the server-side part is not open for public access and the changes applied in CoScripter were only on the client-side, this study doesn't cover the server-side part implementation and characteristics.

Web Navigation Helper (WNH), as previously introduced, is an extension built on top of CS. It was developed basically using CS code and adapting it to attend the new requirements. As such, most of WNH code was developed in the same Firefox add-on technologies mentioned above (XUL, JavaScript, CSS and XPCOM), and in comply with the same software architecture. WNH, however, works not only with those technologies, but with Java as well. The integration of Java to its source code allows new functionalities to be easily implemented. The considerations of using this programming language instead of others were basically because the WNH developer feels more comfortable in developing new code in Java than in JavaScript or any other software development language. Two Java components are currently in use in WNH, one in charge of retrieving a list of all shared scripts of the web site open in the browser window and the other useful for WNH-support, for helping detecting which scripts might be out-of-date. An earlier version of WNH, when plan 'A' was still the primarily focus of this research, had a Java component to generate the script descriptions in natural language. This component is now commented in the code.

In order to better understand the code changes applied in WNH, here is provided a functional description of CS architecture, its main units and modules, as can be seen in Figure 5 (a unit can be understood as a functional perspective resulting from the combination and interaction of two or more modules):

1) Script Commands Builder Module – this module is called by the recorder unit once the demonstration is parsed (the action taken in the web site is identified and interpreted);

2) Script Commands Parser Module – this is where each of the script steps is parsed and interpreted. CS works with a limited number of commands, which can have two arguments at most. Each command is in the form of a sloppy language, which "…*interprets pseudo-natural language instructions (as opposed to formal syntactic statements) in the context of a given web page's elements and actions.*" (Little, et al., 2007). Commands in sloppy language are in the form of human readable text;

3) Decision Making Module – this module receives a parsed script command and the html code and has the not so easy task to eliminate any ambiguity that might come out when trying to identify the elements of the command in the page. The way this module works is by relying upon a heuristic

algorithm. The arguments of the commands are extracted using various heuristics (e.g. has quotes around it, or is followed/preceded by a preposition). (Little, et al., 2007);

4) Html Parser Module – this module is the basis of both Recorder and Execution modules. Since CS records and plays users' interactions, it needs to identify the html page elements;

5) Personal Database Data Retrieval/Storage Module – this is the module in charge of connecting the personal database with the other modules that interact with it. It both stores and retrieves data input from there. Note that the Edit Module makes use of this module in order to create or edit variables in the Personal Database;

6) Script Edition Module – users are able to manually edit the script commands in an edit panel. In this way, either the user might alter an existing script or create a new one. In this module users can also edit data in the personal database;

7) Shared Scripts Finding Module – this attends for the collaboration part of CoScripter. It allows users to see all public scripts related to the current web site;

8) Script Recorder Unit – this unit monitors user's actions such as button clicks, combo box selects, data inputs, etc. in the web site and records each action as a step to be later played back. In order to record a script, CS relies on the Html Parser Module to parse and interpret html code from the current web site and on a builder module, to create a command which reflects the action taken by the user;

9) Script Execution Unit – this unit is in charge of reproducing a sequence of steps. It relies on modules 2-5 to parse the current command, parse the html code, decide what action to take and retrieve any needed data from the personal database;

10) Personal Database – data that is frequently needed during the scripts executions, such as the user social ID number, or birth date, can be

saved in a database. Then, whenever needed, the execution module should retrieve this information from there if available;

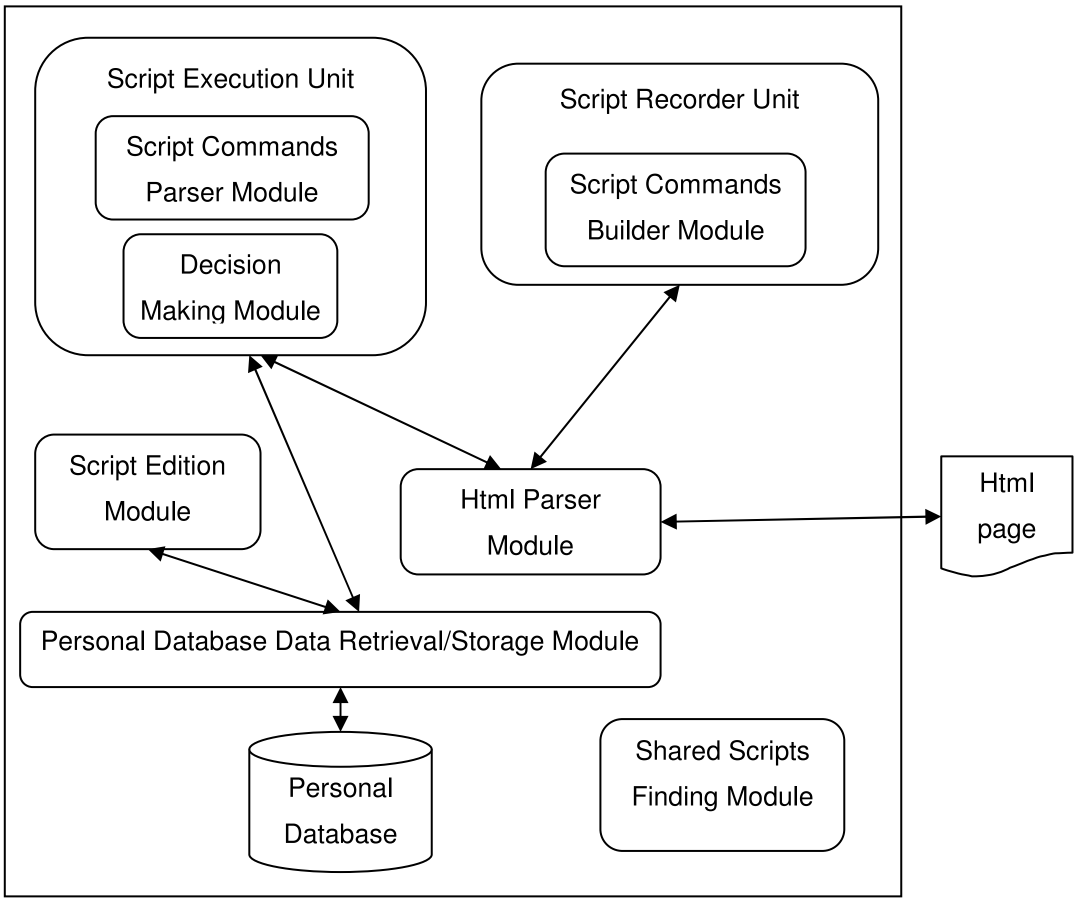

Script Execution Unit
Script Commands Parser Module
Decision Making Module
Script Recorder Unit
Script Commands Builder Module
Script Edition Module
Html Parser Module
Html page
Personal Database Data Retrieval/Storage Module
Personal Database
Shared Scripts Finding Module

Figure 5: CoScripter Architecture

The list below presents the types of modules that can be found in WNH in relation to CS code. While few modules were not modified, some of them were modified during WNH development. The items below describe each type and point to the modules that fit each one of them:

1) Disabled Modules – as stated in chapter 3.2.1, in WNH-see/WNH-read there was a need to disable whatever could mislead the user from achieving the final goal. Therefore, few of these modules were taken away from the interface and turned to be unreachable. Functionally speaking, although the source code is present, it is like these functionalities were stripped of CS when conceiving WNH-see/WNH-read. Figure 6 shows these modules in purple. Since WNH-see/WNH-read end-

users don't need to edit scripts, access or view the personal database nor create any script, the Script Edition Module, Script Recorder Unit, Personal Database and its retrieval/storage module were disabled in WNH-see/WNH-read;

2) Preserved Modules – these are the modules that are still used in WNH and preserve the same behavior as in CS. In other words, they were not changed. They can be seen in Figure 6 in orange. The Decision Making, Html Parser and Shared Scripts Finding modules belong to CoScripter machine, and were needed during script finding and process execution of WNH. Therefore, they were kept unchanged in WNH;

3) Changed Modules – few of the modules were changed by WNH to attend its new requirements. They are pointed out in Figure 6 in green. The Script Execution Unit, which contains the Script Commands Parser Module, was changed in order to accommodate the interception logic of WNH. When each command is parsed, WNH decides how to proceed: if data should be input by users, WNH presents a dialog box asking for it; if a comment is identified, WNH presents a pop-up message box with its contents; and so forth;

4) New Modules – these include the functionalities that were added, colored in beige. The two new modules created in WNH are the Sites Tracker Bot, described in details in section 3.4.3, and the Shared Scripts Description Generation modules. The first one is in charge of aiding support users in maintaining the scripts compatible to the websites in which they take place. The second one was created when plan A was still the main focus of this research. This module would interpret every individual shared script, combine this information with relevant information from the website contents in which it takes place, and extract a summary of the purpose of the script, in natural language, to be presented to end-users. This module is now commented on the code, and should be enabled back when WNH heads back to plan A again;

Figure 6 details these modules in WNH's architecture.

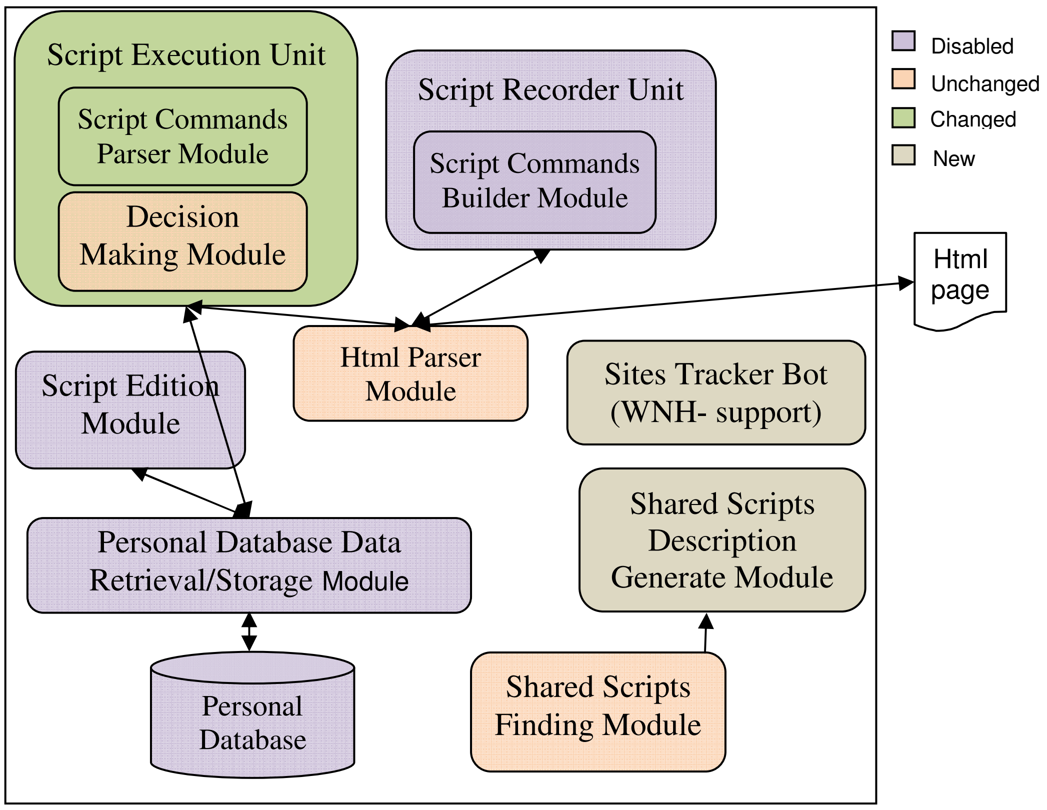

Script Execution Unit
Script Commands Parser Module
Decision Making Module
Script Recorder Unit
Script Commands Builder Module
Html page
Disabled
Unchanged
Changed
New
Html Parser Module
Sites Tracker Bot (WNH- support)
Script Edition Module
Personal Database Data Retrieval/Storage Module
Shared Scripts Description Generate Module
Personal Database
Shared Scripts Finding Module

Figure 6: WNH Architecture

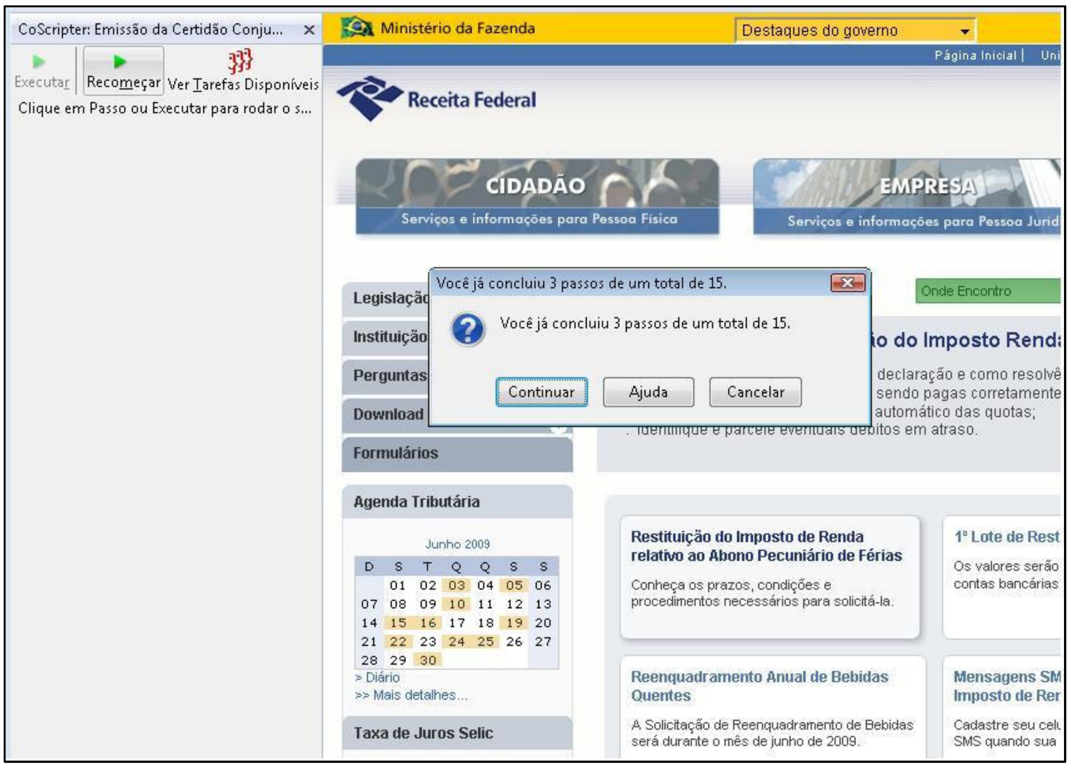
Next section provides illustrations for users interactions with the WNH tool.

3.4. Illustration

This section presents illustrations of the various types of interactions one can have when using the WNH tool. Three different interactions, for three different target users, are considered: 1) Blind users; 2) Functionally illiterate users; 3) Support users.

3.4.1. Blind User Interaction

The figures below present a few excerpts of a blind user interaction in WNH-see. Each action performed in the process execution is called a *step*. Figure 7 shows WNH-see informing the user that step 3 was concluded. This pop-up message is how WNH-see informs at each step the status of the user in the whole process. The need for this message emerged during the first experiments, as will be explained in more detail later on.

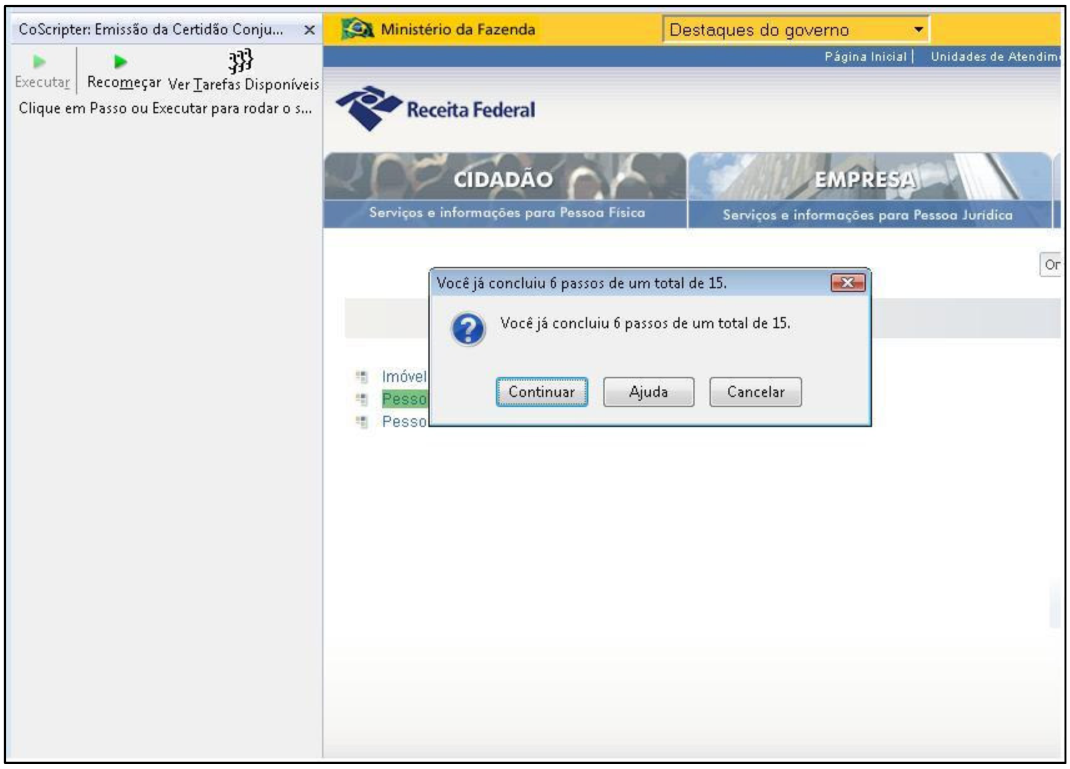


Clique em Passo ou Executar para rodar o s...

Você já concluiu 3 passos de um total de 15.

Figure 7: Step 3 out of 15 (blind users WNH navigation)

In Figure 8, WNH-see informs the user that step 6, out of 15, was concluded.

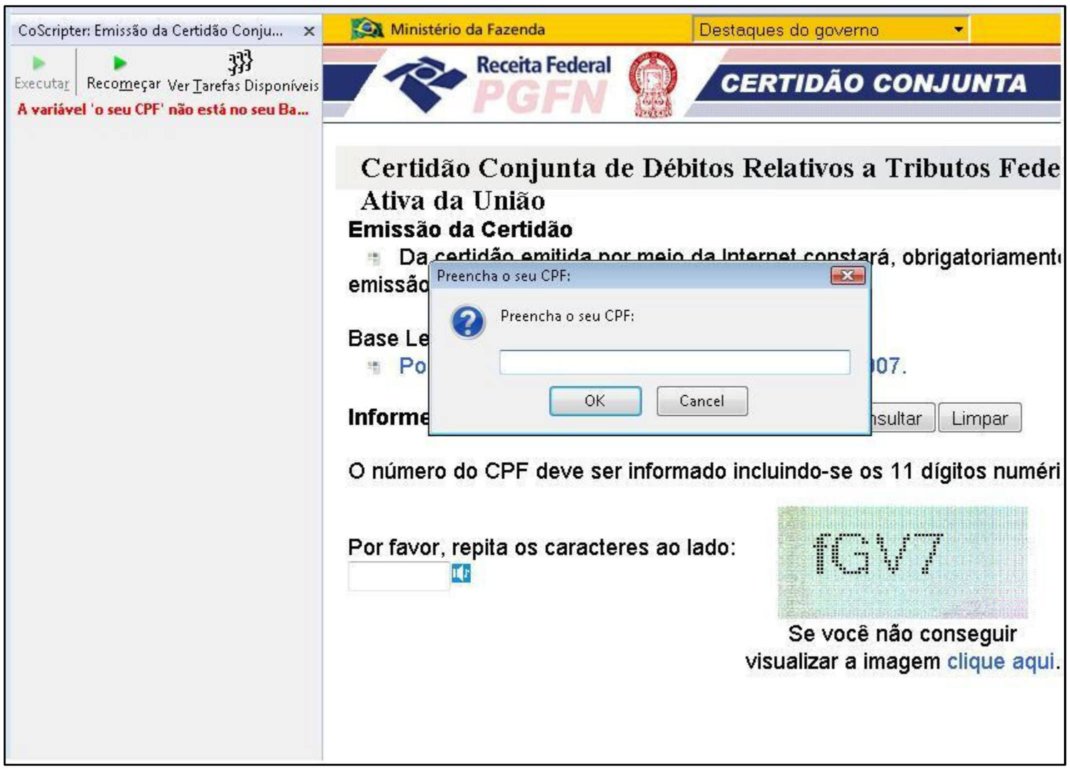


Clique em Passo ou Executar para rodar o s...

Você já concluiu 6 passos de um total de 15.

Figure 8: Step 6 out of 15 (blind users WNH navigation)

Figure 9 shows how WNH-see requests the CPF Id card input from the user.



A variável 'o seu CPF' não está no seu Ba...

Certidão Conjunta de Débitos Relativos a Tributos Fede
Ativa da União

Emissão da Certidão

Da certidão emitida por meio da Internet constará, obrigatoriament
emissão

Preencha o seu CPF:

Preencha o seu CPF:

Base Le

Po 07.

Informe

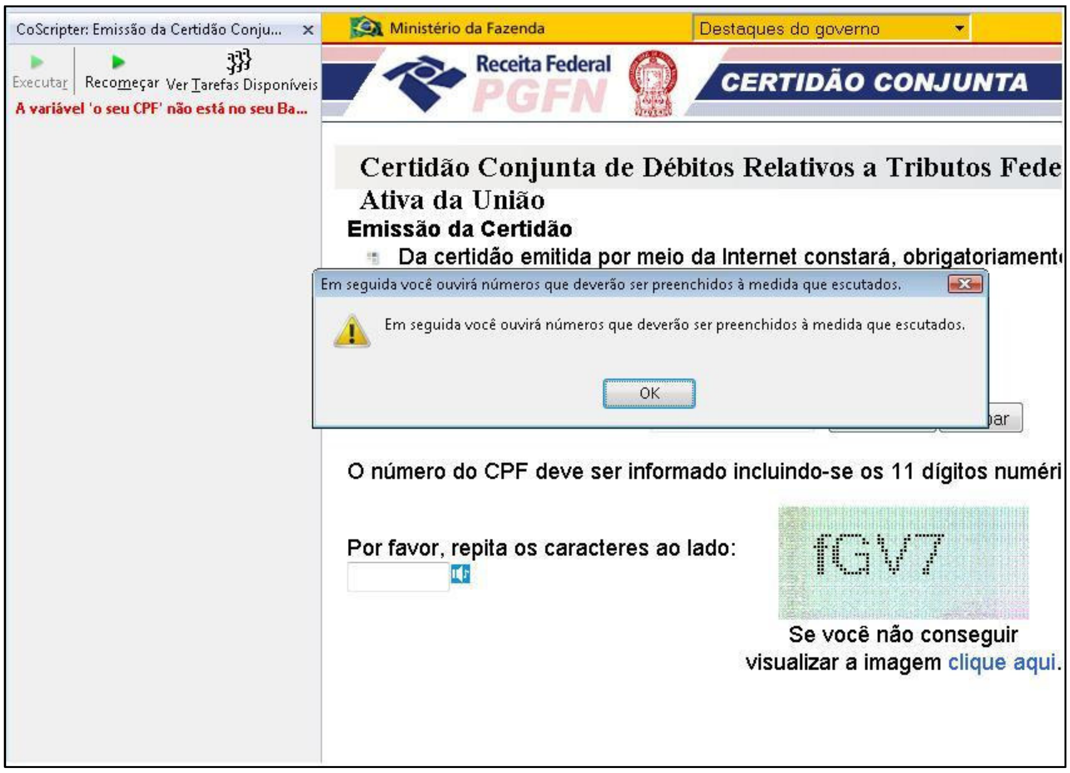
O número do CPF deve ser informado incluindo-se os 11 dígitos numéri

Por favor, repita os caracteres ao lado:

Se você não conseguir visualizar a imagem clique aqui.

Figure 9: Step in which the CPF Id is requested (blind users WNH navigation)

In Figure 10, WNH-see informs the user that the audio captcha numbers will be read out-loud. This comment was introduced in the script execution to allow the end-user to get ready for listening to the captcha and filling it in the input field.


CoScripter: Emissão da Certidão Conju...

Executar Recomeçar Ver Tarefas Disponíveis

A variável 'o seu CPF' não está no seu Ba...

Ministério da Fazenda

Destaques do governo

Receita Federal PGFN

CERTIDÃO CONJUNTA

Certidão Conjunta de Débitos Relativos a Tributos Fede
Ativa da União

Emissão da Certidão

Da certidão emitida por meio da Internet constará, obrigatoriament

Em seguida você ouvirá números que deverão ser preenchidos à medida que escutados.

Em seguida você ouvirá números que deverão ser preenchidos à medida que escutados.

OK

O número do CPF deve ser informado incluindo-se os 11 dígitos numéri

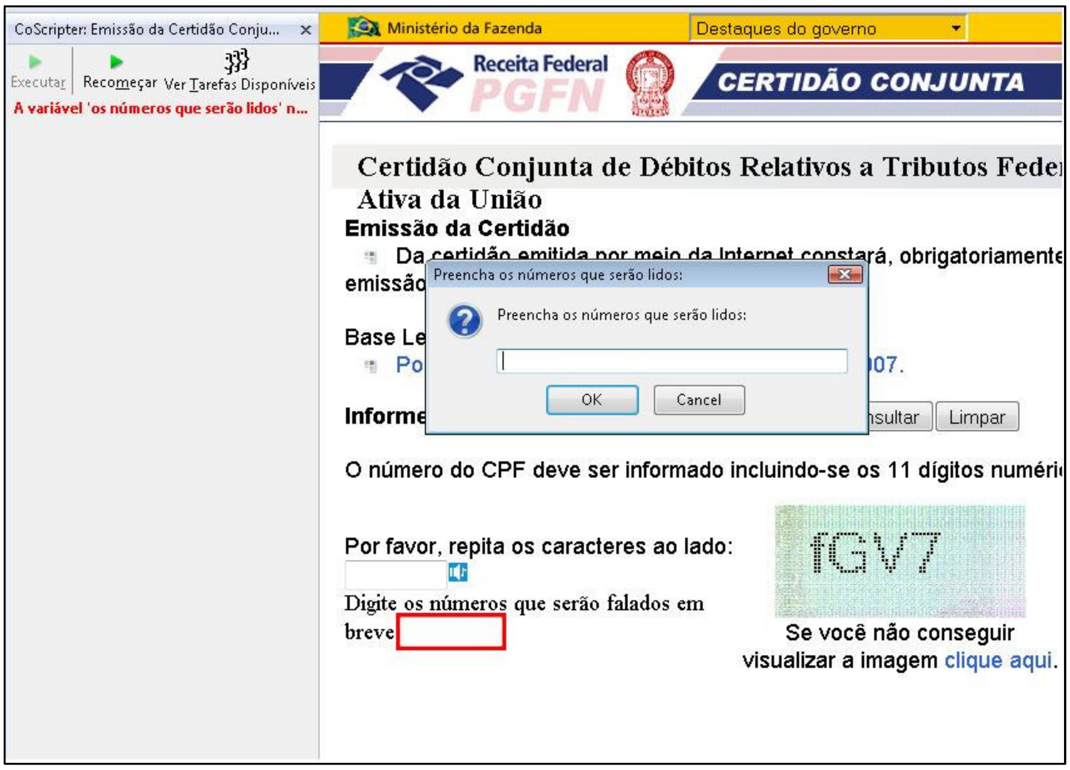
Por favor, repita os caracteres ao lado:

fGV7

Se você não conseguir visualizar a imagem clique aqui.

Figure 10: Step that notifies of the audio captcha (blind users WNH navigation)

Figure 11 shows the moment in which the audio captcha is being read and the input field in which the user should type in the numbers heard.


CoScripter: Emissão da Certidão Conju...

Executar Recomeçar Ver Tarefas Disponíveis

A variável 'os números que serão lidos' n...

Ministério da Fazenda

Destaques do governo

Receita Federal PGFN

CERTIDÃO CONJUNTA

Certidão Conjunta de Débitos Relativos a Tributos Feder
Ativa da União

Emissão da Certidão

Da certidão emitida por meio da Internet constará, obrigatoriamente
emissão

Base Le

Po 07.

Informe

Preencha os números que serão lidos:

Preencha os números que serão lidos:

OK Cancel

nsultar Limpar

O número do CPF deve ser informado incluindo-se os 11 dígitos numéri

Por favor, repita os caracteres ao lado:

fGV7

Digite os números que serão falados em breve

Se você não conseguir visualizar a imagem clique aqui.

Figure 11: Step in which the audio captcha is being read (blind users WNH navigation)

3.4.2. Functionally Illiterate User Interaction

Figure 13 and Figure 14 present two steps of a functionally illiterate user interaction with WNH-read. Figure 12 shows WNH-read requesting for the captcha numbers and characters.

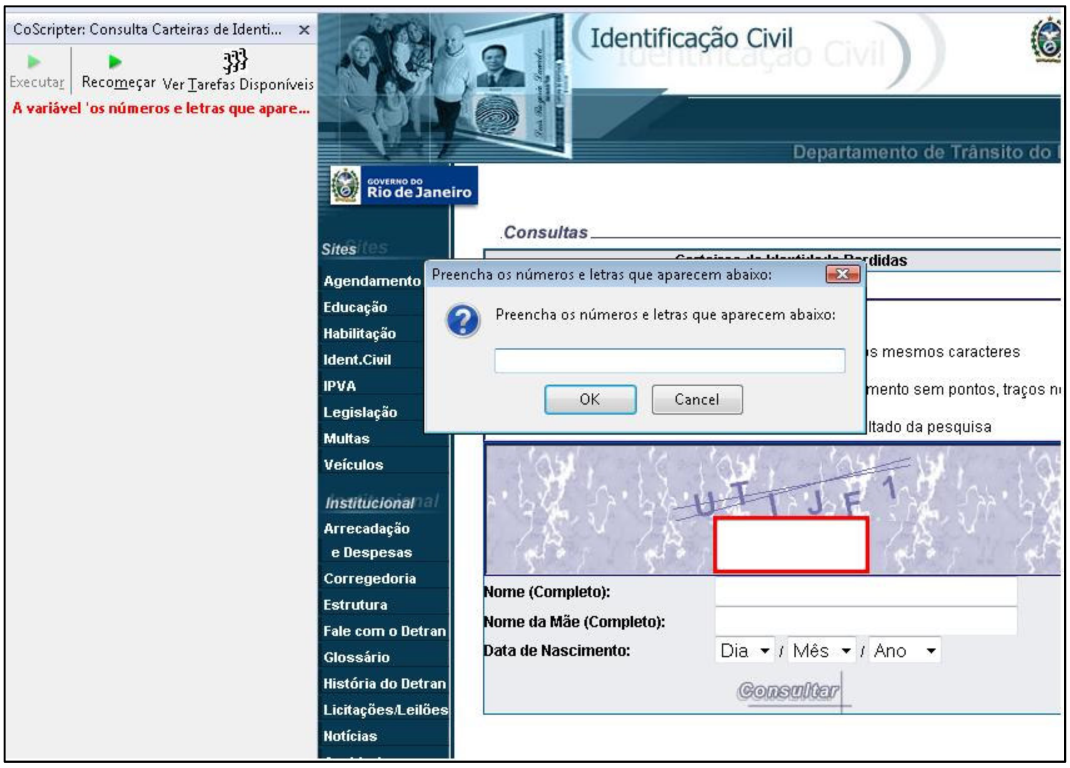
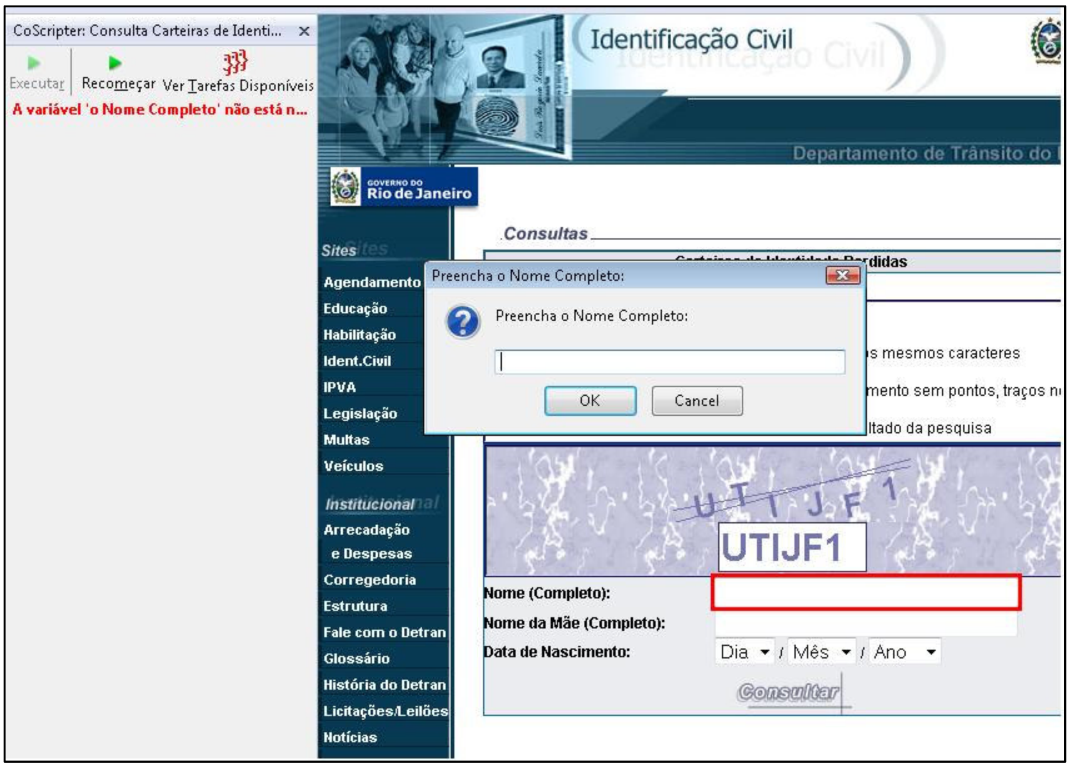


Figure 12: Step that requests the captcha (functionally illiterate users WNH navigation)

Figure 13 shows the request of the complete name data input.



A variável 'o Nome Completo' não está n...

Preencha o Nome Completo:

Preencha o Nome Completo:

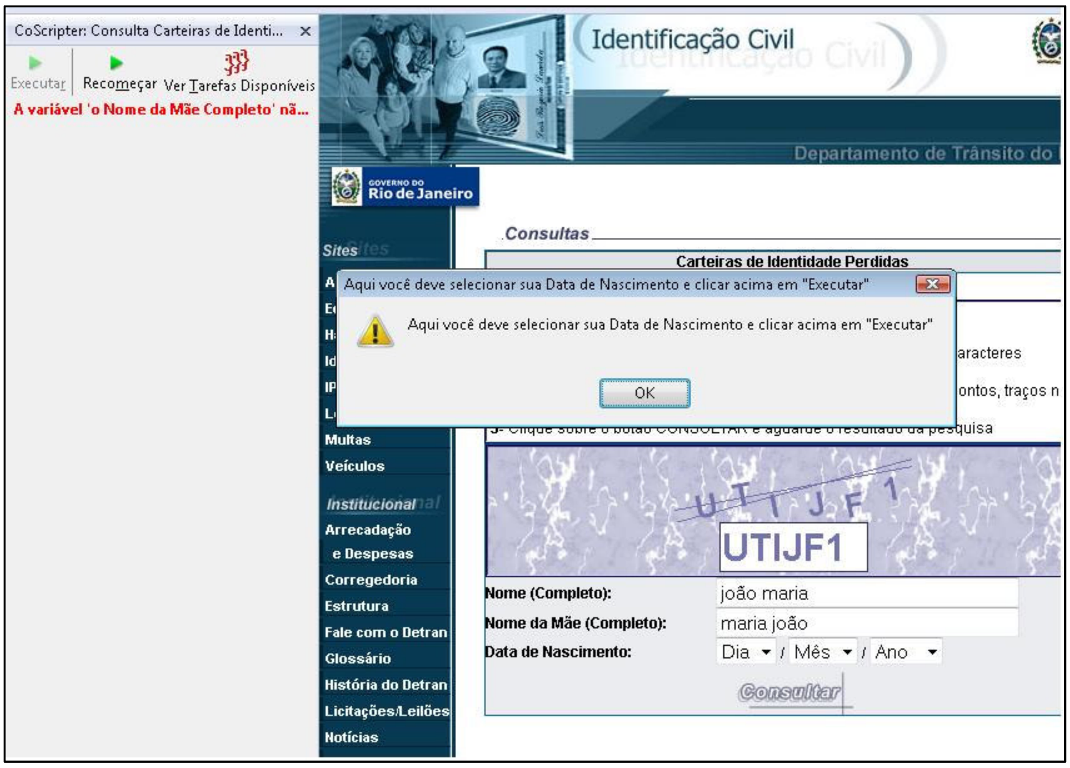
Nome (Completo):

Nome da Mãe (Completo):

Data de Nascimento:

Figure 13: Step that requests the complete name (functionally illiterate users WNH navigation)

Figure 14 shows WNH-read instructing the user to choose the birthday dates in web page, and trigger back the execution of the process by clicking in the "Execute" button.


CoScripter: Consulta Carteiras de Identi...

A variável 'o Nome da Mãe Completo' nã...

Aqui você deve selecionar sua Data de Nascimento e clicar acima em "Executar"

Aqui você deve selecionar sua Data de Nascimento e clicar acima em "Executar"

Carteiras de Identidade Perdidas

Nome (Completo): joão maria

Nome da Mãe (Completo): maria joão

Data de Nascimento: Dia / Mês / Ano

Figure 14: Step that informs the user to fill the birthday directly in the web page (functionally illiterate users WNH navigation)

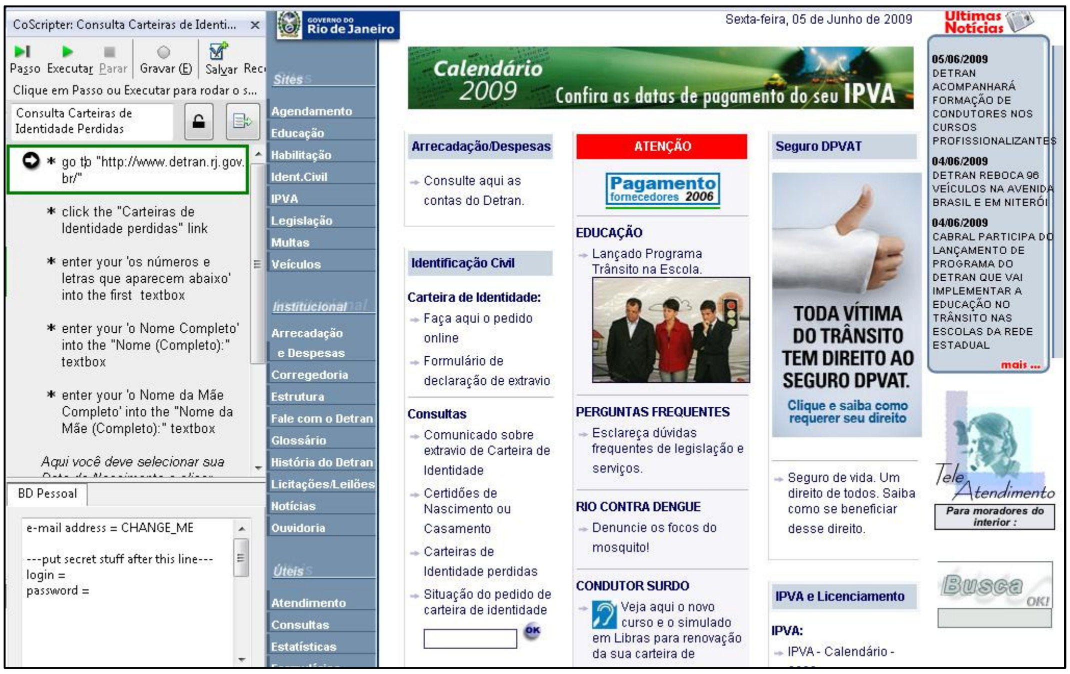
The main differences between the current version of WNH-see and WNH-read interactions are:

1) WNH-see informs at each step the status of the user in the whole process;

2) WNH-see displays the *help* and *restart* buttons at every step during the process;

3.4.3. Support User Interaction

Figure 15 and Figure 16 present two steps of a support user interaction with WNH-support, when executing the script created for the functionally illiterate users' experiment. It is important to notice that the execution is the same. The differences rely on the user interface. Using this version, support users have the full control over the scripts, and can change it if needed.


CoScripter: Consulta Carteiras de Identi...
Passo Executar Parar Gravar (E) Salvar Rec
Clique em Passo ou Executar para rodar o s...
Consulta Carteiras de Identidade Perdidas

* go to "http://www.detran.rj.gov.br/"
* click the "Carteiras de Identidade perdidas" link
* enter your 'os números e letras que aparecem abaixo' into the first textbox
* enter your 'o Nome Completo' into the "Nome (Completo):" textbox
* enter your 'o Nome da Mãe Completo' into the "Nome da Mãe (Completo):" textbox

Aqui você deve selecionar sua

BD Pessoal
e-mail address = CHANGE_ME
---put secret stuff after this line---
login =
password =

Governo do Rio de Janeiro
Sexta-feira, 05 de Junho de 2009
Calendário 2009 Confira as datas de pagamento do seu IPVA
Sites: Agendamento, Educação, Habilitação, Ident.Civil, IPVA, Legislação, Multas, Veículos
Institucional: Arrecadação e Despesas, Corregedoria, Estrutura, Fale com o Detran, Glossário, História do Detran, Licitações/Leilões, Notícias, Ouvidoria
Úteis: Atendimento, Consultas, Estatísticas

Arrecadação/Despesas
Consulte aqui as contas do Detran.

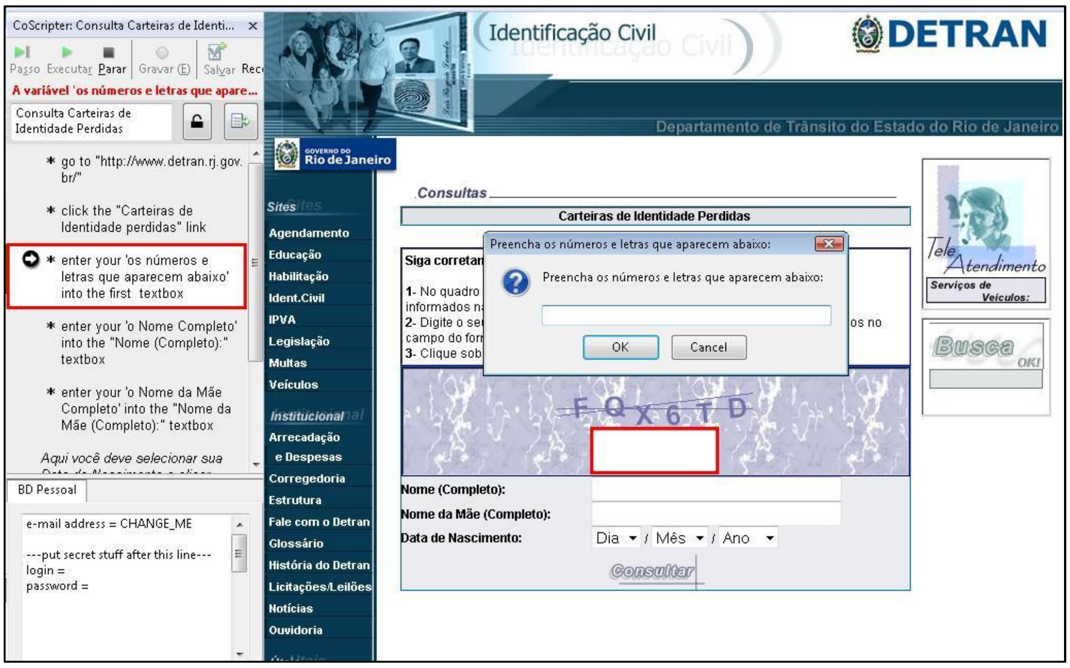
Identificação Civil
Carteira de Identidade:
Faça aqui o pedido online
Formulário de declaração de extravio
Consultas
Comunicado sobre extravio de Carteira de Identidade
Certidões de Nascimento ou Casamento
Carteiras de Identidade perdidas
Situação do pedido de carteira de identidade

ATENÇÃO
Pagamento fornecedores 2006
EDUCAÇÃO
Lançado Programa Trânsito na Escola.
PERGUNTAS FREQUENTES
Esclareça dúvidas frequentes de legislação e serviços.
RIO CONTRA DENGUE
Denuncie os focos do mosquito!
CONDUTOR SURDO
Veja aqui o novo curso e o simulado em Libras para renovação da sua carteira de

Seguro DPVAT
TODA VÍTIMA DO TRÂNSITO TEM DIREITO AO SEGURO DPVAT.
Clique e saiba como requerer seu direito
Seguro de vida. Um direito de todos. Saiba como se beneficiar desse direito.
IPVA e Licenciamento
IPVA:
IPVA - Calendário -

Últimas Notícias
05/06/2009 DETRAN ACOMPANHARÁ FORMAÇÃO DE CONDUTORES NOS CURSOS PROFISSIONALIZANTES
04/06/2009 DETRAN REBOCA 96 VEÍCULOS NA AVENIDA BRASIL E EM NITERÓI
04/06/2009 CABRAL PARTICIPA DO LANÇAMENTO DE PROGRAMA DO DETRAN QUE VAI IMPLEMENTAR A EDUCAÇÃO NO TRÂNSITO NAS ESCOLAS DA REDE ESTADUAL
mais ...
Tele Atendimento Para moradores do interior :
Busca OK!

Figure 15: Volunteers' visualization of step 1 from functionally illiterate's script


CoScripter: Consulta Carteiras de Identi...
Passo Executar Parar Gravar (E) Salvar Rec
A variável 'os números e letras que apare...
Consulta Carteiras de Identidade Perdidas

* go to "http://www.detran.rj.gov.br/"
* click the "Carteiras de Identidade perdidas" link
* enter your 'os números e letras que aparecem abaixo' into the first textbox
* enter your 'o Nome Completo' into the "Nome (Completo):" textbox
* enter your 'o Nome da Mãe Completo' into the "Nome da Mãe (Completo):" textbox

Aqui você deve selecionar sua

BD Pessoal
e-mail address = CHANGE_ME
---put secret stuff after this line---
login =
password =

Identificação Civil
DETRAN
Departamento de Trânsito do Estado do Rio de Janeiro
Consultas
Carteiras de Identidade Perdidas
Siga corretam
1- No quadro informados n
2- Digite o se campo do for
3- Clique sob
os no

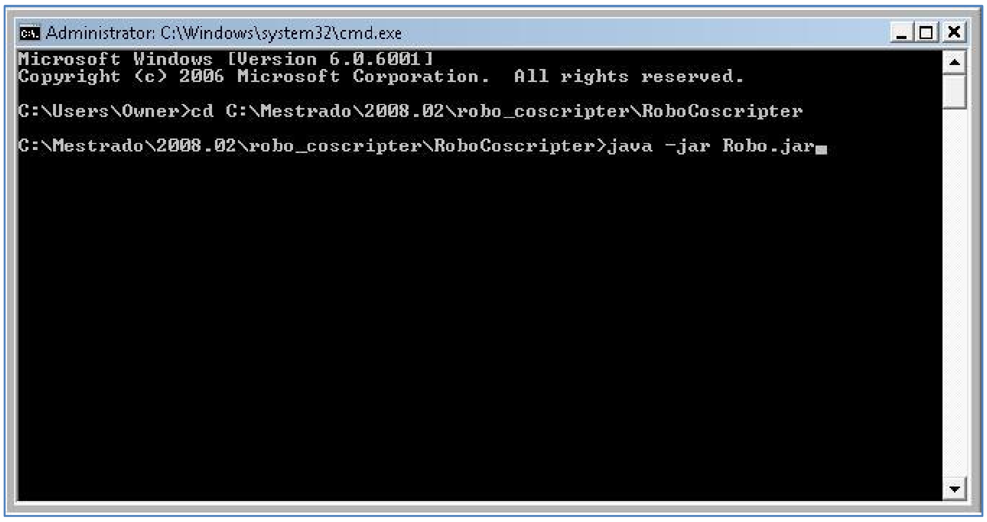
Preencha os números e letras que aparecem abaixo:
Preencha os números e letras que aparecem abaixo:
OK Cancel

FQX6TD
Nome (Completo):
Nome da Mãe (Completo):
Data de Nascimento: Dia / Mês / Ano
Consultar
Tele Atendimento Serviços de Veículos:
Busca OK!

Figure 16: Volunteers' visualization of captcha step from functionally illiterate's script

Volunteers not only create scripts. As mentioned before, changes to web sites are not uncommon, and the scripts might stop working for that reason. It is up to them to keep the scripts active and running. WNH-support has a batch program that should be run daily, from which volunteer users can get the information regarding which scripts ran smoothly and which did not. Those that did not successfully complete are presented in the output list with the status of *suspicious*. These suspicious scripts deserve a closer look, since they might point to possible incompatibilities between the script and the website in which it runs.

WNH-support contains a module called Sites Tracker Bot, written in Java, which is run as a command line from any DOS window in a machine with a Java Virtual Machine installed on, as shown in Figure 17.



```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\Owner>cd C:\Mestrado\2008.02\robo_coscripter\RoboCoscripter

C:\Mestrado\2008.02\robo_coscripter\RoboCoscripter>java -jar Robo.jar
```

Figure 17: WNH-support Sites Tracker Bot command line execution

This bot takes as input a text file containing a list of script names, and outputs two lists, one containing the scripts that ran smoothly, another containing the suspicious scripts. Every script in the input list should be available as separate text files in the local directory. Each one of them will be opened and executed by the bot. The bot then triggers the WNH execution in the Mozilla Firefox browser passing on the information about the location and the name of the script as an argument. In WNH-support mode, either the process completion or its failure is written in the output list, in the local directory. At the end, the volunteers can query the output lists in search of possibly unmatched “script/website” pairs. Each output file contains the date it was ran, so a history of changes can be traced back.

The current bot version handles only scripts that take place in governmental websites, since this is the scope of this dissertation. Further versions should extend this in accordance with WNH.

3.5. Error Prevention, Error Detection and Error Recovery during Interaction

In order to prevent errors, WNH-see makes use of a Help button, which is available at every individual step of the blind users' process execution and guides them through it. After clicking on the help button, users are able to restart the same whole process or start a new one. WNH-read does not provide the help button to its users. Besides the help button, other elements should be incorporated to WNH-read, such as data input validation, and they are left for future work.

WNH has not adopted any particular technique in order to detect the occurrence of errors, which has serious consequences to the user interaction. Evidence of this is when users fill the captcha input field with the wrong value. In this case, the process will get stuck, and it is up to the user to restart it or try a workaround. An elegant solution to that would be, for instance, as soon as the process gets stuck, to offer the user the options of restarting the whole process, or repeating the last step(s), or even starting a new process.

As for error recovery, WNH-see presents the blind users with the option of restarting the whole process or starting a new one. Again, the solution above would be perfectly adequate here: WNH could offer users the same option of repeating the last step(s).

Future versions of WNH should make use as much as possible of different techniques of error prevention, detection and recovery, part of which will be worked out in Future Work.