Better Bioinformatics Through Usability Analysis

Supplementary Information

Davide Bolchini, Anthony Finkelstein, Vito Perrone and Sylvia Nagl
Contacts: davide.bolchini@lu.unisi.ch

Abstract

With this supplementary information, we present additional data concerning the two usability studies that have been carried out to identify and characterize a first set of usability problems of web-based bioinformatics resources. We provide a more complete set of quantitative results from the usability inspection realized on the CATH website (section 1), including the explanation of the inspection process, and summary data about the usability problems found. Section 2 offers an overview of the quantitative results yielded in the user testing focussing on search tasks of three web bioinformatics website, namely BIOCARTA, NCBI and SwissProt.

1. Usability Inspection on CATH

1.1 The process of usability inspection with MILE+

As preparatory activity for the inspection of CATH, it was important to characterize the "object" of the usability analysis. To this end, we have developed a synopsis of the information architecture and navigation capabilities of the application using the IDM method [Bolchini et al., 2006). The map (figure 1) represents the core content topics available to the user, the main navigation connections between these topics and strategies for accessing them. For a detailed reference on IDM concepts and visual notation see [Bolchini et al., 2006). Having a clear abstraction of the information architecture has a number of advantage for usability analysis, including the ability to master the complexity of the information architecture at a glance (and at the appropriate level of detail), and to have a reference map for the ongoing inspection.
For CATH, the high-level modeling of the information architecture resulted in the identification of 4 core content topics: protein domain, chain, PDB description, and homologous superfamily. Each topic presents multiple instances. For example, at the time of the inspection, there were 93,885 protein domains, 63,453 chains, 30,028 PDBs, and 1,459 superfamilies. These topics represent the key target content of the user interaction. To access these topics, CATH offers multiple navigation structures. To access and browse protein domains, CATH provides a primary classification of proteins in 4 main categories (Class 1, Class 2, Class 3 and Class 4). Each main category in turns contains an 8-level hierarchical structure of protein classification (Architecture, Topology, Homologous Superfamily, and 4 further levels related to the sequence similarity criteria). Once the user eventually gets through the hierarchy and reaches the details of a protein domain, s/he can traverse navigation connections to related content: to the corresponding chain or to the corresponding PDB. The access structures to the homologous superfamilies are organised as a full index of all the superfamilies and a summary table. Once in the details of a superfamily, the user can navigate directly to the corresponding superfamily level in the protein classification hierarchy. At the highest level of these access structures there is additional introductory information explaining the purpose and the criteria for the classification of the content.

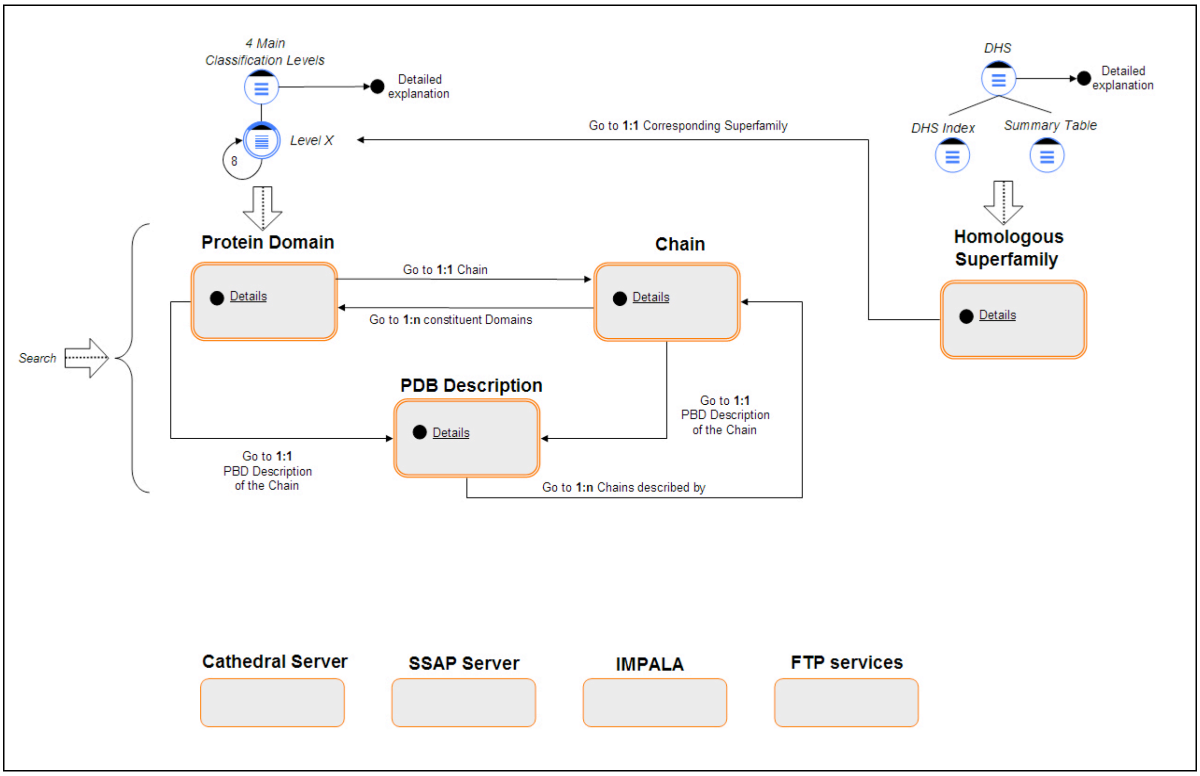

Figure 1. High-level information architecture map for CATH.

A search engine provides keyword-based query access to the topics and more static, ancillary content and upload services are available (upload servers, IMPALA, or FTP services).
These elements, which support the core of the user experience with the applications, constitute the key objects of investigation for the usability inspection and represent the context in which we frame the usability problems.

During the execution of the usability inspection, there are various strategies to employ the set of 82 heuristics offered by MILE+. The choice of the strategy depends on a number of factors including the time available to the inspectors, their experience in usability evaluation, their knowledge of the application domain, and the size of the application (Triacca, 2005).

We can identify 3 main strategies:

- "Leopard's spots" evaluation
- Feature-driven evaluation
- Scenario-driven evaluation

Leopard's spots evaluation or "critical exploration"

This type of evaluation is useful when the application is small or for focussing on specific site sections (to quickly narrow the scope of the analysis). It consists in exploring in depth a section of the application and then randomly moving to other sections, trying to put the application to test with the heuristics at hand. The initial focus is typical on interface and navigation-related aspects; however, technological problems may emerge during exploration, as well as content-related issues. This inspection approach may be called Leopard's spots evaluation because it almost randomly starts from a specific point in the application and then the analysis naturally expands to include other parts on the basis of the criticalities identified. For example, with the MILE+ heuristics at hand, in a preliminary navigation within the main sections of the CATH website, an inconsistency of the presence of the main navigation bar elements (technically called "landmarks") has been spotted. Then, a specific analysis of these landmarks has been carried out and the problem has been identified

and characterized in its various aspects thanks to the MILE+ heuristics. Later on, all other landmarks in other sections have been checked for consistency.
This strategy for MILE+ inspection works very well for a preliminary and quick check-up of the website, and if the inspectors has long-standing usability analysis experience (although not necessarily in the specific application domain) and know the heuristics well.

Feature-driven evaluation
Once inspectors have acquired enough confidence with the application, a more systematic inspection can start. Feature-driven inspection consists in checking each design dimension individually (navigation, content, interface and technology) according to MILE+ heuristics, and organize the inspection process around the applicable features to which heuristics are associated (e.g. guided tour navigation, tree navigation, page layout grid, etc.). This strategy is the most expensive in terms of time but the most rewarding in terms of number of types of problems found. Once a type of problem is found according to a heuristics, the inspector can verify how many instances of that problem occur, to get an additional indicator about the severity of the problem in terms of its pervasiveness in the application. In CATH for example, the usability issues associated with the guided tour navigation feature around which the Dictionary of Homologous Superfamily is organized have been spotted through the navigation heuristics. Then, once identified the rigidity of this specific guided tour, other occurrences of this problem in other access structures have been hunted for.

Scenario-driven evaluation
Scenarios can be used not only during user testing (as described in the paper), but also as drivers for inspection. Building scenarios for usability inspection means identifying a relevant user profile for the web application and envision the way in which s/he may use it in a typical situation to accomplish one or more tasks. To circumscribe the scope of the inspection, a scenario typically focuses on selected sections of the application. As inspectors try to execute the scenarios, usability problems may emerge as obstacles to the successful accomplishment of the scenario.

For example, when focussing on the navigation of the hierarchical (tree-based) classification of protein domains, inspectors tried to reach the details of a protein domain starting from the highest level (class), assuming – after discussion with CATH designers – that is a typical exploratory task to be supported for CATH users. Problems related to the inefficiency of the navigation mechanism in the hierarchy emerged in this analysis process. At each level of the hierarchy, for example, it is shown the total number of sons for that branch. As a matter of fact, the sub-branches at each level are very unbalanced in terms of population. The system, however, does not provide a way to see how all the sons are distributed over the next sub-branches. Therefore, the user cannot be aware of the fact that some sub-branches are very populated (e.g. 100 sons), while others are practically empty (1 or 2 sons). The only way to discover it is by navigating in each sub-branches (going up and down in the hierarchy), which makes the whole exploratory process much more cumbersome and inefficient.

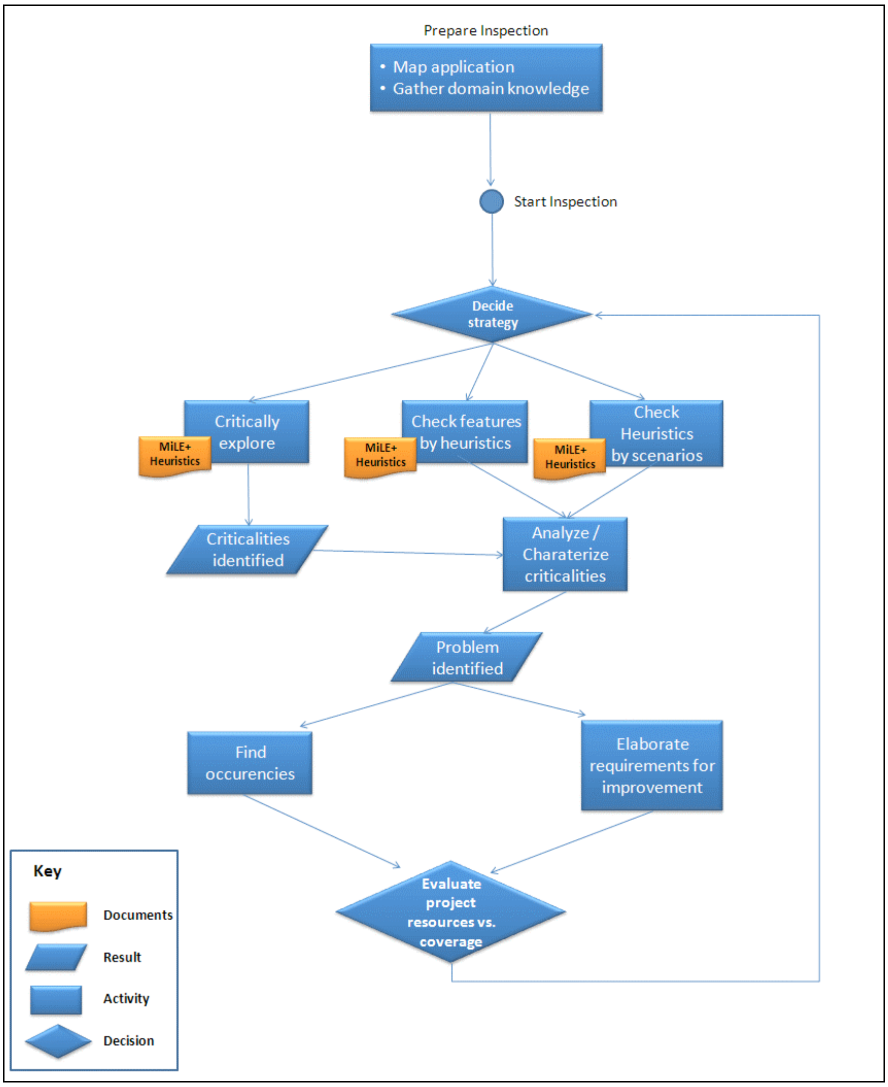

Figure 2. An overview of the usability inspection process with MILE+.

The three strategies can be used in parallel and in a blended way (see Figure 2). For the usability inspection of CATH the inspector started from a preliminary Leopard's spot evaluation, given an initial low confidence and knowledge in the specific CATH domain. As usability issues began to emerge, a feature-driven evaluation was more systematically carried out starting from the critical areas identified. Scenario-driven evaluation was then used for specific application areas, such as the navigation of the hierarchical protein classification, where plausible and typical user's tasks could be easily anticipated, given the nature of the information architecture.

1.2 Summary of Usability Problems

The types of usability problems discovered and addressed during usability analysis of CATH mainly concern navigation and interface design. It is interesting to note that the actual occurrences of these problems often reach a very high number because of the database-driven nature of the web application, as it automatically replicates the same problem for every data instance of the same type.

For example, a wrong navigation link in each superfamily overview page accounts for one problem types, but it will have as many instances as many superfamilies are present in the repository.
The total number of usability problems found and characterized during inspection are organized by design dimensions (Table 1) and listed according to the relevant MILE+ heuristics (table 2).

	Number of problem types	**Number of problem occurrences**
Content	1	1
Navigation	10	>10'000
Interface design	6	16
Technology/Performance	1	>4000
Tot.	*18*	*>14'000*

Table 1. Synopsis of CATH Usability Problems by Design Dimensions.

Heuristics code	**Heuristics Name**	**N. Problem types**	**N. Problem occurrences**	**Severity**
Content_1	Accuracy			
Content_2	Currency			
Content_3	Coverage			
Content_4	Objectivity			
Content_5	Authority	1	1	Medium
Content_6	Consistency			
Content_7	Text Errors			
Content_8	Multimedia Consistency			
Navigation_1	Topic Segmentation			
Navigation_2	Topic Orientation Clues			
Navigation_3	Topic Accessibility			
Navigation_4	Group Introduction			
Navigation_5	Group Orientation Clues			
Navigation_6	Group Accessibility	2	>4000	High
Navigation_7	Transition List			
Navigation_8	Transition Orientation Clues			
Navigation_9	Transition Accessibility			
Navigation_10	Landmarks	2	10	Medium
Navigation_11	Consistency in Overall Navigation	2*	1500	High
Navigation_12	Overall Accessibility			
Navigation_13	Multiple Topic Consistency			
Navigation_14	Multiple Topic Orientation			
Navigation_15	Multiple Topic Segmentation			
Navigation_16	Multiple Topic Accessibility			
Navigation_17	Multiple Group Introduction			
Navigation_18	Multiple Group Orientation			
Navigation_19	Multiple Group Accessibility			
Navigation_20	Tree Orientation			
Navigation_21	Tree Backward Navigation	1	>4000	Medium
Navigation_22	Tree Depth Anticipation	1	>500	High
Navigation_23	Overall "Go Back"			
Navigation_24	Overall History			
Navigation_25	Guided Tour Orientation			
Navigation_26	Guided Tour Control			
Navigation_27	Guided Tour Strategy	1*	1	High
Navigation_28	Guided Tour Topology	1*	1	High
Navigation_29	Index Orientation			
Navigation_30	Index Control			

Navigation_31	Index Strategy			
Navigation_32	Index Topology			
Navigation_33	All-to-All Orientation			
Navigation_34	All-to-All Control			
Navigation_35	All-to-All Strategy			
Navigation_36	All-to-All Topology			
Interface_1	Label Ambiguity	1	1	
Interface_2	Label Overlapping			
Interface_3	Label Genericity/Technicality	1	2	
Interface_4	Information Scent			
Interface_5	Conventionality			
Interface_6	Intuitiveness			
Interface_7	Area Grouping Adequacy			
Interface_8	Position of Importance			
Interface_9	Information Overload			
Interface_10	Scannability			
Interface_11	Page Grouping Adequacy	1	1	Low
Interface_12	Classification Adequacy			
Interface_13	Separation Adequacy			
Interface_14	Mental Map			
Interface_15	Visual Identity			
Interface_16	Chromatic Code	1	1	Low
Interface_17	Background Contrast	1	1	Low
Interface_18	Font Size			
Interface_19	Font Colour			
Interface_20	Font Type			
Interface_21	Text Layout			
Interface_22	Anchor Identity			
Interface_23	Anchor States			
Interface_24	Icon Consistency			
Interface_25	Widgets Consistency			
Interface_26	Position Consistency			
Interface_27	Layout Grid Consistency	1	10	Medium
Interface_28	Layout Conventions			
Interface_29	Homepage Redundancy			
Interface_30	Homepage Layout			
Interface_31	Homepage Animations			
Tech_1	Management of User Error			
Tech_2	Scripting Errors	1	>4000	Medium
Tech_3	Operations Management			
Tech_4	Browser Compatibility			
Tech_5	Plug-ins Required			
Tech_6	Page Download Time			
Tech_7	Media Streaming Optimization			

Table 2. Full set of usability problems found in CATH, organized by the 82 MILE+ heuristics. Blank cells refer to heuristics which were not applicable or which did not yield usability problems.

* Usability problems described and discussed in the paper, section 4.

2. User Testing

2.1. Characteristics of the user sample

10 subjects have been recruited for the user testing focussing on “search tasks” on three repositories (BioCarta, SwissProt and NCBI). As shown in Table 3, the relevant characteristics of the subjects mainly address their level of background knowledge of bioinformatics and genetics in general, their knowledge (in terms of use) about bioinformatics repositories, the familiarity with the three web applications of the study, as well as the number of years of experience in the field bioinformatics and genetics.

	Knowledge about Bioinformatics repositories (1-5)	Years working in Bioinformatics	Knowledge about the specific repositories	Knowledge about Genetics (1-5)	Years working in Genetics	Role	Research Domain
S_1	3	3	**Biocarta**: never used **Swissprot**: extensive use **NCBI:** extensive use	1	1	Phd student	Biochemistry
S_2	5	7	**Biocarta:** never used **Swissprot:** used **NCBI:** used	3	3	Phd student	Bioinformatics
S_3	2	1	**Biocarta:** never used **Swissprot:** used **NCBI:** used	5	5	Senior researcher	Medical research
S_4	1	1	**Biocarta:** never used **Swissprot:** never used **NCBI:** never used	2	1	Master student	Intelligence Systems
S_5	4	3	**Biocarta:** never used **Swissprot:** used **NCBI:** used	3	2	Research Assistant	Bioinformatics
S_6	2	1	**Biocarta:** never used **Swissprot:** used **NCBI:** used	3	2	Phd student	Virology
S_7	2	2	**Biocarta:** never used **Swissprot:** used **NCBI:** used	3	5	Master student	Biotechnologies
S_8	4	3	**Biocarta:** never used **Swissprot:** used **NCBI:** used	4	6	PhD student	Oncology
S_9	2	1	**Biocarta:** never used **Swissprot:** used **NCBI:** used	4	7	PhD student	Biochemistry
S_{10}	3	3	**Biocarta:** never used **Swissprot:** used **NCBI:** used	3	3	PhD student	Bioinformatics

Table 3. The characteristics of the subjects recruited for the user testing.

From the data gathered during recruitment, the set of subjects turned out to be quite well balanced both in terms of bioinformatics knowledge (AV=2.8 in a 1-5 scale, SD=1.22) and number of years of activity in the field of bioinformatics (AV=2.5 in a 1-5 scale, SD=1.84). A slight tendency toward an overall higher level of expertise in genetics knowledge (AV=3.1, SD=1.10) and years of experience (AV=3.5, SD=2.12) is worth noticing.

2.2. **User Testing: Summary of Quantitative Results**

Table 4 shows the data related to whether or not the user completed the tasks assigned to them. The following scale has been used: 1 denotes successful task completion; 0 indicates “task failed” either because the users gave up or they were obviously out of track after an acceptable time threshold of 15 minutes; 0.5 denotes partial completion (with respect to the content of the task assigned).
Given that the purpose of the search-related usability study was to investigate the nature of the usability problems encountered while searching bioinformatics repositories, the data about the effectiveness in supporting user’s tasks should not be read as a comparative evaluation score between the three web applications (BioCarta, SwissProt and NCBI). The data from the user testing were useful to observe recurrent obstacles users found in searching large repositories and to characterize common criticalities in terms of design assumptions, search interface usability and results displaying. The tasks, therefore, were not designed to systematically put the three applications to test and to compare their relative performances.

	Task 1 (biocarta)		**Task 2** (biocarta)		**Task 3** (biocarta)		**Task 4** (swissprot)		**Task 5** (ncbi)	
	Completion	Comments	Completion	Comments	Completion	Comments	Completion	Comments	Completion	Comments
S_1	1	After search term suggestion	0	Wrong repository	1		0	Can’t find telomerase	1	
S_2	0.5	1 gene out of 3	1	Different ontology used	1		0	Picked wrong record	0	Close to completion
S_3	0	Picked wrong record	1		0		0		0	
S_4	1	Wrong spelling	1	By chance	0	Can’t understand the picture	0	Wrong answer	0	Got lost
S_5	0		1		1		0		0.5	Similar asnwer
S_6	1	After search term suggestion	0		0	Wrong search criteria	0	Wrong search criteria	0	Wrong search criteria
S_7	1	After search term suggestion	1		1		0		0	Gave up
S_8	1		1		1		1		0	Wrong search criteria
S_9	1		0		1		0		1	
S_{10}	0		1		1		1		1	
Tot.	**6.5**		**7**		**7**		**2**		**3.5**	

Table 4. Effectiveness in supporting user’s tasks.

Besides the qualitative characterization of the types of usability problems encountered by the users (reported in the paper), the number of query iterations during search task was an interesting finding (Table 5). In our study, a query iteration indicates the number of different search queries performed in the attempt to complete a task. The number of query iteration is one indicator of the level of difficulty in accomplishing the task, which is in turn determined by the design of the system in interaction with the user’s ability to formulate effective queries.

	Task 1	Task 2	Task 3	Task 4	Task 5
S_1	4	3	n/a	3	3
S_2	4	3	1	3	n/a
S_3	4	1	1	2	1
S_4	3	2	2	2	5
S_5	4	n/a	3	1	3
S_6	5	0	3	3	2
S_7	4	5	1	3	3
S_8	3	3	1	3	4
S_9	2	n/a	2	6	2
S_{10}	5	n/a	1	3	3
Average	*3.80*	*2.43*	*1.67*	*2.90*	*2.89*
St. Deviation	*0.92*	*1.62*	*0.87*	*1.29*	*1.17*

Table 5. Number of search query iterations per task.

Various external factors have influenced these results, however, which cannot be only accounted for the design of the system and the search ability of the user. First of all, the formulation of the task itself may have in some cases tipped off the user by suggesting “ready to use” keywords for the query. These cases might partially account for the low number of query iterations in Task 3. In other cases, the formulation of the task may have suggested terms which are commonly used in the field, but not necessarily supported as they are by the system. In these cases, the user needed to reformulate the query several times to learn the lexicon and syntax supported by the repository search mechanism.

References

Bolchini, D., Paolini, P., Interactive Dialogue Model: a Design Technique for Multi-Channel Applications, IEEE Transactions on Multimedia, 8 (3) 2006, 529-541.

Triacca, L., Inversini, A., Bolchini, D., Evaluating Web Usability with MiLE+, in IEEE Proc. Seventh International Symposium on Web Site Evolution (WSE 2005), Budapest, September 2005, 22-29.