CEUR Make GUI – A usable web frontend supporting the workflow of publishing proceedings of scientific workshops

Muhammad Rohan Ali Asmat[1,3] and Christoph Lange[2,3]

[1] RWTH Aachen, Germany m.rohan.a.asmat@gmail.com
[2] University of Bonn, Germany math.semantic.web@gmail.com
[3] Fraunhofer IAIS, Sankt Augustin, Germany

Abstract. CEUR-WS.org is a widely used open access repository for computer science workshop proceedings. To publish a proceedings volume there, workshop organisers have to follow a complex, error-prone workflow, which mainly involves the creation and submission of an HTML table of contents. With ceur-make we had previously provided a command-line tool for partially automating this workflow. However, in a recent usability evaluation we confirmed that the tool is difficult to learn, highly dependent on other software, not portable and hard to use. We sought to solve these issues with a web-based user interface, which we present here. A usability evaluation of the latter proves significant improvements.

Keywords: Scholarly publishing, open access, user experience, user interfaces

1 Introduction

Scientific social networks such as ResearchGate and free-of-charge open access repositories such as the Computing Research Repository (CoRR[4]) have significantly lowered the barrier for sharing research results in the form of individual papers. Open access repositories for complete proceedings of scientific events include the Proceedings of Machine Learning Research (PMLR) and the Electronic Proceedings in Theoretical Computer Science (EPTCS), addressing specific fields of computer science, and the CEUR Workshop Proceedings (CEUR-WS.org), addressing workshops from all over computer science.[5] Each of these employ an individual workflow for publishing, which proceedings editors and/or authors need to follow strictly to keep the effort low for those who run the service, usually volunteers. For example, PMLR requires editors to provide a BibTeX metadata database following specific rules[6], EPTCS acts as an overlay to CoRR, i.e. requires papers to be pre-published there, and CEUR-WS.org requires editors to

[4] `http://arxiv.org`
[5] `http://proceedings.mlr.press/`, `http://www.eptcs.org/`, `http://ceur-ws.org`
[6] `http://proceedings.mlr.press/spec.html`

provide an HTML table of contents following a certain structure[7]. Here, we focus on facilitating the latter by adding a web-based graphical user interface to a tool that auto-generates such tables of content, improving over the usability issues of the previous standalone command-line version of that tool.

Section 2 provides a more precise problem statement. Section 3 discusses related work. Section 4 presents the design and implementation of our web-based user interface. Section 5 evaluates the usability of the frontend in comparison to its command-line backend. Section 6 concludes with an outlook to future work.

2 Problem Statement

2.1 The Publishing Workflow

The HTML table of contents of a CEUR-WS.org workshop proceedings volume includes metadata about the workshop (title, date, venue, proceedings editors, etc.) and each of its papers (title, authors). This structure is prescribed[8]; around once a year, it has so far evolved a bit, e.g., in the form of more explicit semantic annotations to facilitate reuse of the metadata. Besides following the latest template and producing syntactically valid HTML, requirements for proceedings editors include following a consistent capitalisation scheme for paper titles, providing full names of authors, and using relative links to the full texts of the individual papers (typically PDF files). The HTML table of contents together with the full texts has to be submitted to CEUR-WS.org as a ZIP archive.

2.2 Automation of the Workflow with ceur-make

Traditionally, proceedings editors had to prepare the submission ZIP file manually. With ceur-make[9], the second author, technical editor of CEUR-WS.org, has provided a tool to automate part of this job – aiming at three objectives:

– Helping proceedings editors to learn more quickly how to create a table of contents, reducing their effort, and helping recurrent editors to cope with structural changes.
– Reducing the workload of the volunteers who carry out the subsequent publishing steps at CEUR-WS.org; so far, around one in ten submissions requires further communication with its editors to resolve problems, mainly rooted in the table of contents.
– Reducing the implications that subsequent improvements to the structure of the table of contents have on both proceedings editors and the CEUR-WS.org team by reducing their exposure to manual editing.

[7] `http://ceur-ws.org/HOWTOSUBMIT.html#PREPARE`
[8] `http://ceur-ws.org/Vol-XXX/`
[9] `https://github.com/ceurws/ceur-make`

For ceur-make, the metadata about the workshop and its papers have to be provided in two XML files. ceur-make can auto-generate the latter XML file from the metadata that the widely used EasyChair submission and review management system exports in LNCS mode (cf. Section 3.1). From these two XML files, ceur-make auto-generates an HTML table of contents and finally a ZIP archive conforming with the CEUR-WS.org requirements. In addition, ceur-make can generate a BibTeX database to facilitate the citation of the papers in a proceedings volume, as well as a copyright form by which the authors agree to the publication of their papers with CEUR-WS.org.

2.3 Shortcomings of ceur-make

Shortcomings of ceur-make include that it depends on a Unix-style command line environment and a number of software packages that typically only developers have installed: the Make build automation tool[10], the Saxon XSLT processor and the Perl scripting language. Furthermore, it requires proceedings editors to edit one or two XML files manually, without validating their content with regard to all rules that editors should follow. It also requires them to follow certain conventions for naming and arranging files and directories; most importantly, the sources of ceur-make have to be downloaded to the same directory in which the proceedings volume is being prepared. These reasons may explain why ceur-make has so far only been used for less than one in ten proceedings volumes.

2.4 Research Objectives

The objectives of our research were 1. to assess the shortcomings of ceur-make in a more systematic way, and 2. to overcome them by providing a user-friendly web frontend to ceur-make.

3 Related Work

3.1 Conference Management Systems

The complex process of managing scientific events (conferences, workshops, etc.) is facilitated by a broad range of systems, of which we briefly review three representatives and their proceedings generation capabilities. Parra et al. have reviewed further systems without providing details on proceedings generation [6]. In computer science, **EasyChair**[11] enjoys particularly wide usage.[12] EasyChair features a special "proceedings manager" interface, which is initialised by adding all accepted papers and then supports the collection of the final ("camera ready") versions, including a printable form (usually PDF), editable sources (LaTeX,

[10] `https://www.gnu.org/software/make/`

[11] `http://www.easychair.org`

[12] EasyChair has so far been used to manage 53,739 events and has had 1,954,080 users (`http://www.easychair.org/users.cgi`, accessed 2017-04-18).

Word, or anything else, e.g., HTML, in a ZIP archive), and a copyright transfer form. Proceedings chairs can define an order of papers and add or edit additional documents such as a preface. Specific support for exporting all these files and their metadata is provided for events that publish in Springer's Lecture Notes in Computer Science (LNCS) series. Microsoft's **Conference Management Toolkit** (CMT[13]) assists with publishing accepted papers to CoRR. With a professional license, **ConfTool**[14] supports the export of metadata in multiple formats (Excel, XML and CSV) to facilitate proceedings generation.

3.2 Usability Evaluation of Command Line vs. GUI

Comparing the usability of command-line (CLI) vs. graphical user interfaces (GUI) has been a long-standing research topic. Hazari and Reaves have evaluated the performance of students in technical writings tasks using a graphical word processor vs. a command-line tool [3]. Starting from the same level of background knowledge and given the same time for training, a significantly larger share of users felt comfortable using the GUI rather than the command line; also, their task-based performance was slightly higher with the GUI. Gracoli is an operating system shell with a hybrid user interface that combines GUI and CLI [12]. Its design is motivated by common drawbacks of CLIs, which are stated as follows: – the user can interact with the application in a limited way; – the output is hard to understand for the user; – the user does not easily get a clue of how to perform a task.

4 Design and Implementation of CEUR Make GUI

4.1 Architecture

The CEUR Make GUI is a graphical layer built on top of ceur-make. Figure 1 shows its three-layer architecture (Interface, Middleware, and Storage).

The **Interface Layer** consists of all the presentation elements. It displays visual elements, handles dependencies on external libraries for user interface elements, styles the web pages, validates forms and manages user interaction the web pages. It also initiates the communication with the Middleware Layer on user's request and displays the results from the Middleware. Technologies used on Interface Layer include standard web technologies used for front end clients (HTML 5, CSS, JavaScript), and the following libraries: Materialize CSS[15] is a JavaScript and CSS library based on Google's Material Design principles[16], used here to incorporate standard design patterns into the GUI. jQuery Steps[17] is used to create wizards for taking inputs.

[13] `https://cmt.research.microsoft.com/cmt/`
[14] `http://www.conftool.net/`
[15] `http://materializecss.com`
[16] `https://material.io/guidelines/`
[17] `http://www.jquery-steps.com`

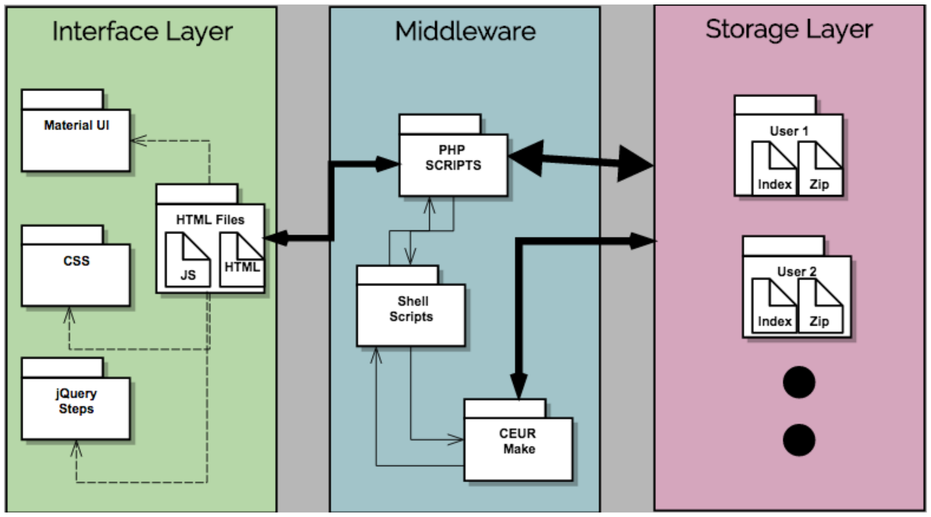


Fig. 1: System Architecture of CEUR Make Graphical User Interface

The **Middleware Layer** generates artifacts required for publishing at CEUR-WS.org. The Middleware Layer creates the files, as requested through the Interface Layer, by running ceur-make. It returns links to the artifacts stored at the Storage Layer to the Interface Layer, thus acting as a service provider.

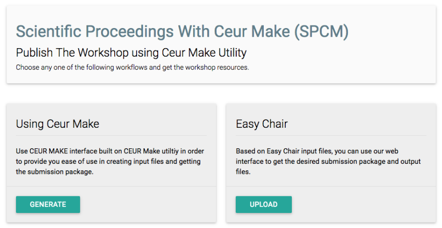
The **Storage Layer** stores the files that are created temporarily on the server. It separates the files based on the user's identity and then also based on the workflow that the user chooses to create the artifacts for publishing (manual metadata input vs. EasyChair import).

4.2 Interface

Fig. 2: Navigational Menu of CEUR Make Graphical User Interface

We aim at providing an easy to use, task oriented interface. On the main screen, we give users the option of switching between four tasks: *viewing announcements*, *viewing published proceedings*, *publishing a proceeding* and *reporting an issue* through a navigational menu (cf. Figure 2). Further, we separate the site navigation of the two proceedings publishing workflows using a Card design pattern [9], representing each workflow as an independent card (cf. Figure 3a). We follow the Wizard design pattern [11] to collect workshop metadata input from users (cf. Figure 3b). For the list of all proceedings volumes[18], we follow the style of the CEUR-WS.org user interface but make it more accessible by following standard design patterns. We applied the Pagination design pattern [10] to address the problem of the current CEUR-WS.org site that one has to scroll down a lot because all content is displayed at once; secondly, we applied the Autocomplete design pattern [8] to facilitate the task of finding proceedings volumes already published at CEUR-WS.org easier.

Source code, documentation and a working installation of the CEUR Make GUI are available at `https://github.com/ceurws/ceur-make-ui`.


Scientific Proceedings With Ceur Make (SPCM)

Publish The Workshop using Ceur Make Utility

Choose any one of the following workflows and get the workshop resources.

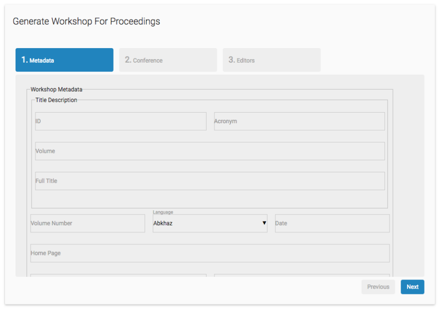
Using Ceur Make

Use CEUR MAKE interface built on CEUR Make utiltiy in order to provide you ease of use in creating input files and getting the submission package.

Easy Chair

Based on Easy Chair input files, you can use our web interface to get the desired submission package and output files.

(a) Workflows for Publishing Proceedings


Generate Workshop For Proceedings

(b) Workshop metadata input wizard

Fig. 3: Workflow Screens

5 Evaluation

5.1 Methodology

Participants Twelve persons participated in the evaluation of the usability of the ceur-make CLI vs. the GUI. We chose nine participants with previous CEUR-WS.org publishing experience[19] and three participants without. The latter were trained to publish at CEUR-WS.org to avoid learning biases in our evaluation results.

Procedure The participants were divided into two groups based on their availability. Those who were physically available participated in a Thinking Aloud test [7], and the other ones participated in a Question Asking test:

[18] For now, we only implemented the list of proceedings volumes as a hard-coded mockup for the purpose of evaluating the usability of our user interface design.

[19] Among them we would have preferred to have some with ceur-make experience, but this proved infeasible given the small number of people who had ever used it.

Thinking Aloud: Participants were provided with task definitions as explained below. They were asked to think aloud about their plans and their interaction with the system, particularly including problems they faced or unusual mental models, while the evaluator took notes. The task completion time was recorded for the purpose of comparison.

Question Asking: In a video conferencing setting with screen sharing (using Skype), the evaluator performed each task according to its definition. The participants were allowed to ask questions during the usability test, where the evaluator also asked questions to test the user's understanding. From an audio recording, the evaluator compiled a transcipt of pain points afterwards.

Following a within-subject design setup[20], each participant first had to test the CLI and then the GUI. The participants were given four tasks to be performed in each system, designed to cover all major use cases of the system in a comparable way: 1. Initiate generation of a proceedings volume, 2. Generating workshop metadata, 3. Generating table of contents metadata, and 4. Search a proceedings volume. Each tasks were subdivided into smaller steps, e.g., as follows for Task 4:

Task 4 – Search a Proceedings Volume

1. Go to the proceedings page at `http://ceur-ws.org` (or in the GUI, respectively).
2. Search the proceedings volume that has the name "Cultures of Participation in the Digital Age 2015".

For the full list of task definitions, please see appendix A and B of [1].

Usability tests were followed by a post study questionnaire for each user, which was created and filled using Google Forms[21]. The questionnaire was divided into following sections:

System Usability Scale (SUS [4]), a ten point heuristic questionnaire to evaluate general usability of the system on a Likert scale from 1 (strongly agree) to 5 (strongly disagree).

Question for User Interaction Satisfaction (QUIS [5]), a 27 point questionnaire to evaluate specific usability aspects of the system, covering overall reaction to the software, screen layout, terminology and system information, as well as learning and system capabilities, from a scale from 0 (lowest) to 9 (highest). The mean score was calculated for every user.

Dataset All users used the same input data for both systems to ensure unbiased comparability of the content created and of completion times across users and systems. A full record of the data is provided in appendix A and B of [1].

[20] `https://web.mst.edu/~psyworld/within_subjects.htm`

[21] `https://www.google.com/forms/about/`

5.2 Results

This section summarizes the evaluation results; for full details see appendix C and D of [1].

Table 1: Quantitative Usability Evaluation Results using Thinking Aloud

Tasks	**CEUR Make (Minutes)**	**CEUR Make GUI (Minutes)**
Task 1	0.13	0.10
Task 2	4.77	2.88
Task 3	2.40	1.46
Task 4	0.76	0.10

Quantitative Results (Completion Times) Table 1 shows the completion times per system and task – in detail:

1. **Task 1 (Initiate generation of a proceedings volume):** This required entering a terminal command for the CLI and pressing a button in the GUI. On average, this took less time in the CLI, but the difference is too marginal to be significant.
2. **Task 2 (Generating Workshop Metadata):** This required entering workshop metadata into the GUI input wizard, and using a text editor and the command line in the CLI. The difference in completion time is significant: completing the task using the GUI took only 60% of the time taken using the CLI, which emphasizes the user-friendliness of the GUI.
3. **Task 3 (Generating Table of Contents Metadata):** This required entering metadata of two papers similarly as for Task 2, with similar results.
4. **Task 4 (Search a proceedings volume):** This task took 7.6 times as long on the CEUR-WS.org homepage compared to the GUI. This result highlights the importance of using the autocomplete design pattern for searching in the graphical user interface, compared to just the "find in page" search built into browsers.

Overall, users took significantly less time to complete tasks with the GUI, which proves the usability improvement it provides over the CLI.

Qualitative Results Notes recorded while performing the usability test were categorized in ten heuristics, i.e.: ***Speed*** *in performing a task*, ***Documentation*** *of the software*, *Ease in performing a* ***Task***, ***Learnability*** *of the software*, *clear* ***Navigation*** *structure of the system*, ***Portability*** *of the system*, ***Error*** *correction chances*, *easy to use* ***Interface***, ***Dependency*** *on other systems* and ***Features*** *to be added.* Table 2 shows the number of responses of the twelve

Table 2: Qualitative Results for ceur-make and the CEUR Make GUI

Heuristics	**ceur-make**	**# of Responses from Users**	**CEUR Make GUI**	**# of Responses from Users**
Speed	Good	3	Neutral	–
Documentation	Good	4	Neutral	–
Task	Good	8	Neutral	–
Learnability	Bad	5	Good	5
Navigation	Bad	4	Good/Bad	5 / 2
Portability	Bad	2	Good	2
Error	Bad	2	Good	3
Interface	Bad	4	Good	8
Dependency	Bad	4	Good	3
Feature	Neutral	–	Excited	6
Total Responses		36		34

participants for each qualitative heuristic, where "bad" means they were not comfortable using it, "good" means they liked the software, and "excited" means that the user is interested but would like to see more features to be implemented.

The total number of qualitative responses was 36 for the CLI and 34 for the GUI. 15 good responses were recorded for the CLI, regarding the heuristics *Speed*, *Documentation* and *Task*, whereas 21 bad responses were recorded regarding the heuristics *Learnability*, *Portability*, *Navigation*, *Error*, *Interface* and *Dependency*. For the GUI no response was recorded against the heuristics *Speed*, *Documentation* and *Task*, which were reported as good for the CLI. This was the case because users did not require documentation to operate the GUI as they never requested for it from the evaluator, speed was not an issue while using it as they never complained about it, and it enabled them to perform their respective tasks. 26 good responses were recorded for the GUI against the heuristics *Learnability*, *Portability*, *Navigation*, *Error*, *Interface* and *Dependency*, which were all recorded as bad in case of the CLI. This highlights the usability improvement provided by the GUI over the CLI. Only two bad responses were recorded for the GUI, against the heuristic *Navigation*, which means a slight improvement in navigation is required – as quoted by a user: *"ceur-make make things easier but has a complex setup, whereas the GUI is straightforward and requires no prior learning. With little improvement in the flow of screens it could be even better."*

Moreover, for the GUI, 6 responses were recorded as excited, against the heuristic *Feature*, which means users would like to use the software and would like additional features to be integrated.

Post Evaluation Questionnaire The overall usability of the two systems was evaluated using **System Usability Scale**[22]. The SUS score for ceur-make was 41.25, which is below grade F (x-axis) as shown in Figure 4. This rating demands

[22] `http://www.userfocus.co.uk/articles/measuring-usability-with-the-SUS.html`

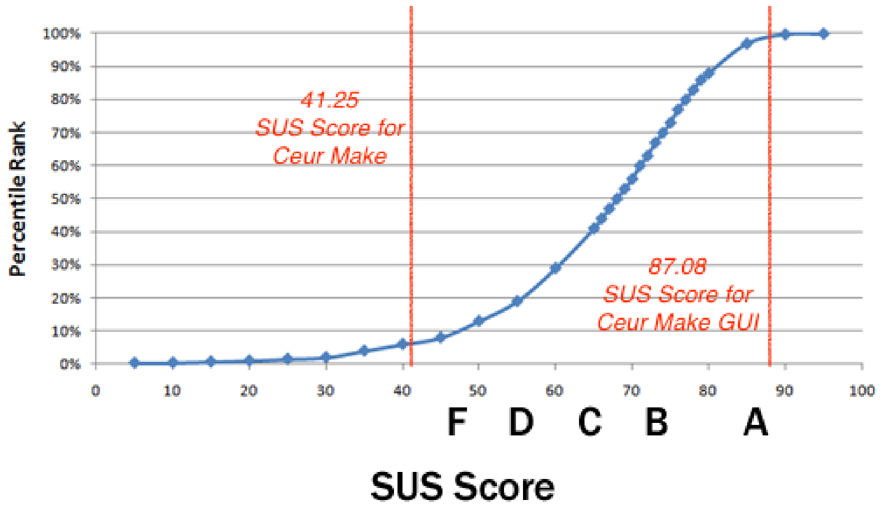


Fig. 4: SUS Score: ceur-make vs CEUR Make Graphical User Interface

immediate usability improvements. On the other hand, the SUS score of the GUI was 87.08, which is above grade A (x-axis). This means that the GUI has a good usability and its users would recommend it to others.

Results of the **Questionnaire for User Interaction Satisfaction** reflect a high usability improvement of the GUI over the CLI. For the questions related to the *learnability of the system*, a visible difference in mean scores was recorded for *easy to remember the commands*, *learning to operate the system* and *trying new features by trial and error*. For these three questions, mean scores of the CLI were 3.75, 3.25, and 4.25 (all below average) and for the GUI they were 8.5, 8.25, and 8.0 (all above average). Likewise, mean scores for the GUI for the questions related to *information representation*, including *information organization*, *positioning of messages*, *highlighting of information to simplify task*, *prompts and progress* were 7.75, 8.0, 6, 6, and 6.25 (above average), whereas for the CLI they were 4, 4, 2.25, 4, and 3.25 – i.e. a notable difference. Another highlight was that users appreciated that the GUI was considered to be *designed for all levels of users* as backed by a mean score of 7.75, whereas the CLI was considered not to be designed for all levels of users as its mean score was just 3.

6 Conclusion

We aimed at automating a workflow for publishing scientific results with open access, focused on the CEUR-WS.org workshop proceedings repository. We developed a graphical user interface on top of the ceur-make command line tool and systematically evaluated the usability of both. Quantitative results on task completion time prove that the GUI is more efficient in performing common tasks. Qualitative evaluation suggests that on all heuristics where ceur-make performed badly, i.e., *learnability*, *navigation*, *portability*, *error*, *interface* and *dependency*, the GUI yielded good responses. In the post-evaluation questionnaires, a notable difference was recorded in the SUS scores of the two systems: grade F for ceur-make vs. grade A for the GUI. 11 out of 27 QUIS questions of ceur-make had responses below average, and others were satisfactory, whereas for the GUI all responses were above average. Overall, results indicate that the usability of the GUI has noticeably improved over the command line. As our evaluation setup covered most typical tasks of proceedings editors, the results suggest that the GUI makes the overall process of publishing with CEUR-WS.org more effective and efficient and thus will attract a broad range of users. Thanks to the input validation of the metadata wizard and to the detailed explicit semantic RDFa annotations of tables of contents that ceur-make outputs, broad usage of the GUI will improve the quality of CEUR-WS.org metadata, largely eliminating the need for reverse-engineering data quality by information extraction (cf. [2]).

Future Work The next immediate step is to officially invite all CEUR-WS.org users to use the GUI for preparing their proceedings volumes. Partly inspired by feedback from the evaluation participants, we are planning to enhance the GUI with functionality addressing the following use cases (all filed as issues at `https://github.com/ceurws/ceur-make-ui/issues/`): **User Profiles** would help to automatically suggest information related to the editors while working on that section of the workshop metadata (Issue #1). Even without user profiles, building and accessing a **database** of previously published workshops' metadata would facilitate input, e.g., by auto-completing author names, and by reusing metadata of a workshop's previous edition. The RDF linked open data embedded into ceur-make generated tables of contents or extracted from old tables of contents (cf. [2]) can serve as such a database once collected in a central place (#5). A **Collaborative Space for Editors** would support multiple editors to work in parallel on a proceedings volume (#2). **Saving System State** would improve user experience and give users more control (#4). Currently, there is no way to restore the state of the interface, where one left in case the browser is accidentally closed or users want to complete the task later. **Extraction of Author Names, Titles and Page Numbers** from the full texts of the papers would further lower task completion time, as the system would automatically suggest most metadata (#3).

References

1. Asmat, M. R. A. A Usable Web Frontend for Supporting the Workflow of Publishing Proceedings of Scientific Workshops. MA thesis. RWTH Aachen, 2016. `http://eis-bonn.github.io/Theses/2016/Rohan_Asmat/thesis.pdf`.
2. Dimou, A. et al. Challenges as enablers for high quality linked data: Insights from the Semantic Publishing Challenge. In: PeerJ Computer Science (2017): *Semantics, Analytics, Visualisation: Enhancing Scholarly Data.*
3. Hazari, S. I., Reaves, R. R. Student preferences toward Microcomputer user Interfaces. In: Computers & Education 22 (3 Apr. 1994).

4. Measuring Usability with the System Usability Scale (SUS). MeasuringU. 2017. https://measuringu.com/sus/ (visited on 2017-04-17).
5. Norman, K. L., Shneiderman, B. Questionnaire for User Interaction Satisfaction QUIS. http://www.lap.umd.edu/quis/ (visited on 2016-09-25).
6. Parra, L. et al. Comparison of Online Platforms for the Review Process of Conference Papers. In: *CONTENT*. 2013.
7. Thinking Aloud: The Number 1 Usability Tool. Nielsen Norman Group. 2017. https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/ (visited on 2017-04-10).
8. User Interaction Design Pattern Library: Autocomplete. UIPatterns. 2017. http://ui-patterns.com/patterns/Autocomplete (visited on 2017-04-17).
9. User Interaction Design Pattern Library: Card. UIPatterns. 2017. http://ui-patterns.com/patterns/cards (visited on 2017-04-17).
10. User Interaction Design Pattern Library: Pagination. UIPatterns. 2017. http://ui-patterns.com/patterns/Pagination (visited on 2017-04-17).
11. User Interaction Design Pattern Library: Wizard. UIPatterns. 2017. http://ui-patterns.com/patterns/Wizard (visited on 2017-04-17).
12. Verma, P. Gracoli: a graphical command line user interface. In: *CSCW*. 2013.