 The definitive version of this paper has been published in Eighth International Conference on Quality of Multimedia Experience (QoMEX), 2016, DOI: 10.1109/QoMEX.2016.7498938

ERWIN - *E*nabling the *R*eproducible Investigation of *W*a*I*ti*N*g Times for Arbitrary Workflows

Thomas Zinner*, Matthias Hirth*, Valentin Fischer*, Oliver Hohlfeld §
* University of Wuerzburg, Chair of Communication Networks, Wuerzburg, Germany
§ RWTH Aachen University, Chair of Communication and Distributed Systems, Aachen, Germany

***Abstract*—Delay effects can impact the Quality of Experience of interactive systems, which motivates research assessing delay impairments, mostly for web based systems. Current studies follow individual methodologies and typically assesses individual and custom-made web pages, whose construction requires expert knowledge in web technologies. Further, a range of native, non-web applications cannot be easily modified for delay studies. Thus, a generalized methodology for assessing delay impacts for a broad range of applications that is accessible to researchers without (web) development expertise is still missing. This paper contributes to this open problem by *i)* presenting a new methodology for reproducible delay assessments in a broad class of systems and *ii)* presenting an open-source implementation to be used by the community. This methodology particularly aims at making delay assessment available to a broad range of researchers by avoiding programming skills and thus by lowering the barrier for setting-up delay assessments.**

I. Introduction

Delay can impair the usage experience of interactive systems and thus represents a relevant class of quality aspects. Yet, the study of delay impairments can be challenged by the broad range of available systems and methodological challenges such as injecting delays into existing systems. Based on their widespread use and possibility for modifications by domain-experts, web based systems are a widely studied class of systems. The widespread use of web based systems, however, requires adequate methods to understand their Quality of Experience (QoE) and to subsequently allow their improvement.

The QoE assessment of interactive—and in particular web based—systems focuses on the study of loading delays or waiting times. Delay can—depending on its severity and the interaction type—impact the user interaction with a system, prolong task completion times, and ultimately lower the QoE. The assessed delay impacts largely focuses on web site loading delays as assessed in controlled but small-scale user studies (see e.g., [1]–[4]) or as assessed in open but large measurements of web systems (see e.g., [5]–[8]). The latter further quantified monetary effects of delay such as loss of revenue [8], [9] or lowered system interactions [7], [9], [10].

Despite these research efforts, no standardized methodology for assessing the impact of delay on a broad class of applications exists. This fact is rooted in a number of challenges. *i)* Existing systems cannot always be modified to test different delay conditions. A particularly interesting class of such unmodifiable systems includes e-commerce and enterprise systems that received less attention in current research. For the limited class of network based applications, network emulators can be applied to inject delays at network level. While these emulators can provide a working solution, the injected delays must be precisely controlled [11] and the conducted studies tend to be non-reproducible since live systems can change during the test. Further, in the absence of a deep system understanding (e.g., due to missing source code) the effects of the injected network delays can lead to unpredictable system behavior and can thus render the obtained results inconclusive. *ii)* Even if systems can be modified, expert knowledge is required to perform the modifications. For example, JavaScript based manipulation of a web page loading behavior (e.g., as used in [4], [12]) requires knowledge in web development and is thus not accessible to non-experts. As a result, performing such tests is limited to researchers with particular domain knowledge, while a broader accessibility would be desired to lower the barrier for performing such research. We therefore posit that a generalized evaluation methodology to assess the effects of delays on the experience of a broad range of interactive systems is required.

In this paper, we aim at closing this gap by describing a generalized methodology for assessing delay effects in a broad range of (interactive) systems. The goal of this methodology is to *i)* lower the barrier for conducting delay assessment studies by avoiding the need for expert knowledge in system development and *ii)* to allow for reproducible studies. We further make our methodology available to the community by providing an open-source implementation that can be used for lab as well as crowdsourcing studies. By this we aim at making a generalized test methodology broadly available to stimulate further research. The contributions of this paper are as follows.

- We propose a widely applicable method for assessing delay effects in a broad range of applications, including web and enterprise applications. The proposed method allows both *i)* the construction of test cases by using traditional means of web development or *ii)* to visually construct test cases of application screenshots and UI overlays. Since the latter does not require any expert knowledge in web development, it makes test construction available to a broad audience of researchers and further ensures repeatability.
- We exemplify our proposed method in an online web testing framework that will be made publicly available.
- We apply our methodology to web QoE as widely used use case to show its general applicability. The obtained results of the performed crowdsourcing study highlight that our methodology does not bias the results as compared to traditional methods for web QoE assessment.

II. Background and Related Work

Loading delays and waiting times denote a class of quality indicators that is particularly relevant for interactive systems. In this regard, a large body of research focuses on understanding delay impairments in the context of web-based systems. This trend is motivated by the fact that web based interfaces are becoming a widespread interface solution for a wide range of applications, ranging from classical web services to enterprise applications. The web browsing experience is mainly impacted by two *known* classes of quality indicators [1]. One class resembles visual properties such as the time of the first visual sign of progress [13]. The second class relates to loading delays and loading pattern, which is in the scope of this paper.

A. Impact of Waiting Times on Interaction

Early indications for the experience of interactive systems were provided by HCI studies on tolerable delays for user interface interaction. These found users to tolerate short delays of up to 0.1 sec for immediate feedback, delays of up to 1 sec for maintaining flow, and delays of up to 10 sec for keeping users focused on the current task [14]. The work on tolerable delays is complemented by studies on neuronal processes and response times. In this regard, Miller [15] observed a 2 secs delay threshold before the information processing in the brain is interrupted. Similarly, [16] finds that frequent users demand response times of less than 1 sec and that productivity increases as response time decrease. The above mentioned works provide general guidelines for interface and interaction design.

B. Delay Perception of Web Pages

A much larger and more recent body of work concerns assessing the quality impact of web page loading delays. In this regard, recent work suggests a logarithmic relationship between subjective quality and page load time (PLT) [1], [2], [17] that is reflected in ITU-T Recommendation G.1030 [18]. The G.1030 Web-QoE model particularly focuses on information retrieval scenarios like web search.

Additional factors have been found meanwhile that are not captured by the G.1030 model. Strohmeier et al. [3] showed that task-driven interaction is judged more critical than task-free interaction. This observation introduced the notion of *relevant elements* to Web-QoE assessment, i.e., elements needed by a user to solve a given task. In addition, it was found that the temporal position of a relevant element influences QoE, leading to better quality ratings when loaded earlier. For task-driven usage the task completition time (TCT) outperforms PLT for quality prediction [3], which could also be shown for multiple consecutive page requests [4]. In an extension, Guse et al. [19] found situational factors to have no significant impact on quality judgment (e.g., driving in public transport vs. laboratory setting) whereas distraction using a parallel task leads to slightly better judgments. These works on web page loading delays are complemented by a study assessing the impact of (partial) load failures [12]. In the presence of partial load failures (i.e., a set of of web page components such as CSS or images loads delayed or is never loaded), the PLT and TCT as currently used quality indicators have been shown to be insufficient to capture the experience quality. Beyond delay, the impact of bandwidth fluctuations and outages has been quantified for web QoE [20].

Web-QoE research is complemented by work in psychology studying acceptable webpage loading delays [21]. This study focused on the impact of a progress indicator on the amount of time until users aborted the load process and reloaded the webpage. It was found that the average waiting time without progress indicator decreased from 13 sec to 3.3 sec from the first to the third non-loading webpage whereas with progress indicator subjects waited first 38 sec and on the third encounter 6.7 sec.

C. Economic Impact of Delays

A number of studies outline the economical relevance of low delays for web browsing [7]–[10]. An increase in latency reduces user interactions with web sites and thus reduces revenue. For example, an increase in the processing delay of 100 ms (400 ms) reduced the daily amount of Google searches per user by 0.2% (0.4%), respectively [9], [10]. Schurman reports similar findings for Microsoft Bing and reports a revenue decline of 0.6% (2.8%) for 500 ms (1000 ms) of additional latency [9]. Sefanov reports a 5% to 9% traffic drop for a latency increase of 400 ms at Yahoo [7]. Linden found that a 100 ms increase in latency drops the sales rate at Amazon by 1% [8]. These examples highlight that even small delay increases can have a significant business impact.

D. Waiting Times for E-Commerce Applications

Pioneer works [22], [23] on e-Commerce identified excessive download delays as major concern. Investigated delays of up to 30 seconds in [22] indicate, that download delay is a major problem second only to security. [23] highlights the impact of anchoring processes, e.g., waiting time messages with approximated waiting durations, on the perceived waiting time and the assessed web page quality. Based on the results, the authors conclude, that waiting times are perceived as intolerable, customers tend to leave the web page and purchase products elsewhere. Further, design guidelines for software developers are derived. A study on key website factors in e-business is conducted in [24]. Besides non-technical impact factors like popularity, usability and website content the authors also identify the download speed as key influence factor for an successful e-business. Details on delay thresholds, however, are not provided by this study.

Synthesis: With respect to the above mentioned related work the following conclusions can be drawn:

- From the point of psychological research, it takes at least one, probably two seconds until waiting delays are perceived as interruption, i.e., opinion scores should be high for these delays. Delays larger than 10 seconds are expected to result in a low user ratings.
- A large body of research focused on assessing web page loading delay of web pages. E-commerce or enterprise applications have been so far neglected. Thus, future delay studies should be assessing a broader range of applications.
- Testing *live* systems is a challenging task and can easily yield *non-reproducible* results since live systems tend to change during the test session.
- User studies are typically of small scale, follow individual methodologies and assess individual and custom-made

web pages, whose construction requires expert knowledge in web technologies. A range of native, non-web applications cannot be *easily* modified for delay studies. Thus, a generalized methodology for assessing delay impacts for a broad range of applications that is assessable to researchers without (web) development expertise is still missing.

III. Methodology for Assessing Delay Impacts

To address the previously outlined challenges, we propose a generalized methodology for assessing delay effects in a broad range of (interactive) systems. The goal of this methodology is to *i)* lower the barrier for conducting delay assessment studies by avoiding the need for expert knowledge in system development and *ii)* to allow for reproducible studies. By this we aim at making a generalized test methodology broadly available to stimulate further research. The requirements of this methodology are as follows.

- To be accessible by a broad range of scientists, its application should not require any domain knowledge (e.g., in web technologies to modify web pages) for testing desired application workflows subject to arbitrary delays.
- The generated test conditions should be easily integrable in existing test platforms, e.g., crowdsourcing platforms to scale-out to a large number of test subjects.
- The generated test conditions should be storable in a persistent manner along with their configuration, e.g., to allow archival, documentation, and repeatability of the generated tests.

Our methodology comprises of mapping the system behavior to workflows, i.e., snapshots of system behavior assessed in the subjective test. For the generation of these workflows (test conditions), we support two modes. In the first mode (specific to web applications), we allow for a traditional test creation by specifying HTML code. As this baseline mode captures the current way or web tests, it suffers from the above mentioned challenges of requiring domain knowledge in test creation and is limited to web application. This motivated us to propose a second mode that is the key of our methodology. This mode comprises of overlaying a number of system screenshots with invisible user interface elements (e.g., buttons and text boxes) to interact with otherwise static screens. The utilized screenshots show the unmodified system user interface and can be captured by any researcher without development expertise. This method further enables us to test a broad range of non-web systems (e.g., SAP enterprise systems) as web app, e.g., to allow crowdsourced tests that would otherwise not be feasible. The system workflow can be represented as a sequence of captured screenshots overlayed with UI elements that can be assigned with individual actions. This allows us to link between the steps of the process. The action triggered by each of the UI fields can be delayed resulting in response delays as desired. The resulting impaired workflow is placed on a webserver and easily accessible by a large number of subjects. Further, it can be integrated in online surveys like surveymonkey.com or TheFragebogen.de. Persistent storage of the overall survey is possible by archiving on the webserver.

We inject a configurable rendering delay into the generated workflow. This allows the study of user-perceived delays in a controlled manner. For the sake of simplicity, we omit the explicit configuration of (implicitly included) other delay sources such as network, input or processing delay which are eventually perceived as rendering delays.

IV. Introducing Delays into Arbitrary Processes

Current web technologies support feature-rich web applications, like word-processor, video editing or even cloud-based operation systems. This enables us to build very detailed mock-ups of user-interfaces (UIs), but this is time-consuming and can become rather complex. However, in most test settings analysing the impact of delays, it is not required that the full functionality of the application is available to the test participant. Thus, a trade-off needs to be made between the level of detail of the mock-up to create an immersive experience of the application and the required implementation effort.

The implementation of our methodology provides means to building detailed UI mock-ups in a timely manner, based on screenshots of existing web applications. Further, it allows to delay UI responses to user inputs to emulate loading or processing delays. To enable large scale QoE test, the resulting mock-up is realized as a web page, so that it can be easily made available to a large group of participants.

Our tool aims at creating mock-ups for two main use-cases, the evaluation of delays in business application and in online web applications. The business applications we consider in the later evaluation are not web based. Consequently, it would be required to rebuild the existing UI in HTML, CSS, and JavaScript, or equivalent technologies to generate a full-feature mock-up. However, this is usually not doable for complex applications with a reasonable amount of effort. Therefore, our tool allows building mock-ups based on screenshots of the application. In contrast, the UI of web applications can directly be integrated into mock-ups designed with our tool. This allows including dynamic effects, like hovering effect of links, which is not possible in the screenshot based approach. However, in order to control the delayed responses to the users' interactions it might be necessary to slightly adapt the HTML code of the original web application, e.g., remove links, and the JavaScript code, e.g., remove OnClick events. Still, both the HTML and the screenshot-based mock-ups have limitations. In both cases, the realization of dynamic content adaptations (e.g., via Ajax) is limited. Unlike in the HTML mode, dynamic content such as videos or animations and web-layouts scaling to arbitrary display sizes cannot be realized with screenshots.

Figure 1 shows a screenshot of the developed mock-up tool. To make it widely accessible, it is implemented as web application which can easily deployed on a server. The source code can be obtained from GitHub[1].

In a first step, the screenshots of the offline application, respectively compressed archive of web application's fronted UI are uploaded to the tool. Then, the screenshot and the HTML pages can be imported into the mock-up working frame (1) of the tool. The content of this frame is later shown to the test participants. UI elements that can be used in the mock-up are shown on the left side next to the working frame. These elements can simply be dragged it the working frame and adapted in size and functionality. In this example a button (2)

[1] https://github.com/lsinfo3/erwin

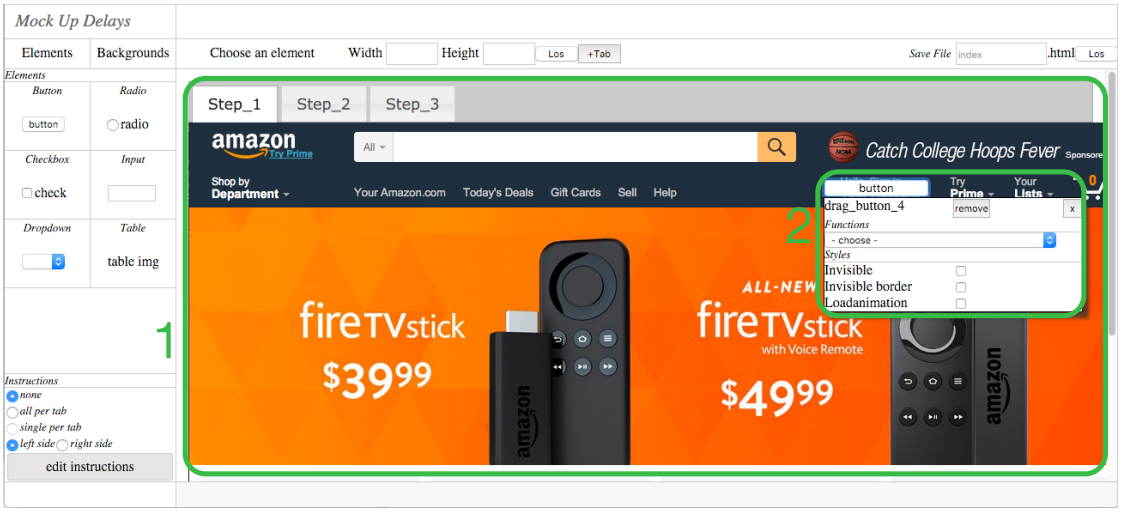


Fig. 1. Mock-up tool

is placed over the login element of the original web page. The tool offers a simple drop down menu to assign different actions to each of the added UI elements. Available actions for the button in this case are displaying a dialog box or forwarding to the next step in the emulated workflow. Moreover, the style attributes of the button can be changed so that it is no longer visible for the test participant.

The given example shows only a first step of a potential test scenario, i.e., creating a delayed action when clicking on the login button. The following steps, e.g., the display of the login form asking for the credentials can be created in a similar manner, by creating a screenshot or downloading the HTML code, uploading it to the mock-up tool, and placing appropriate UI elements. To further support long lasting and complex workflows, the tool also allows adding instructions for the participant that are displayed while proceeding in the test. Here the experimenter can choose to either show the same instructions at each step or display specific instructions, depending on the current progress of the test participant.

After configuring the test workflow, a single HTML file containing the CSS and JavaScript code of the mock-up can be downloaded. While the screenshots increase the overall page size, we avoid the otherwise prolonged loading times by prefetching screenshots with JavaScript. To this end all data is downloaded to the device of the test participants as soon as the test is started (i.e., the size of the screenshots does not impact the assessed delays). The mock-up can be easily deployed on any web-server by placing the downloaded mock-up HTML file, the screenshots or HTML pages, and the publicly available jQuery library into one folder. Only a common web browser with enabled JavaScript is required at the participants' device.

V. Exemplary Use Case – Amazon

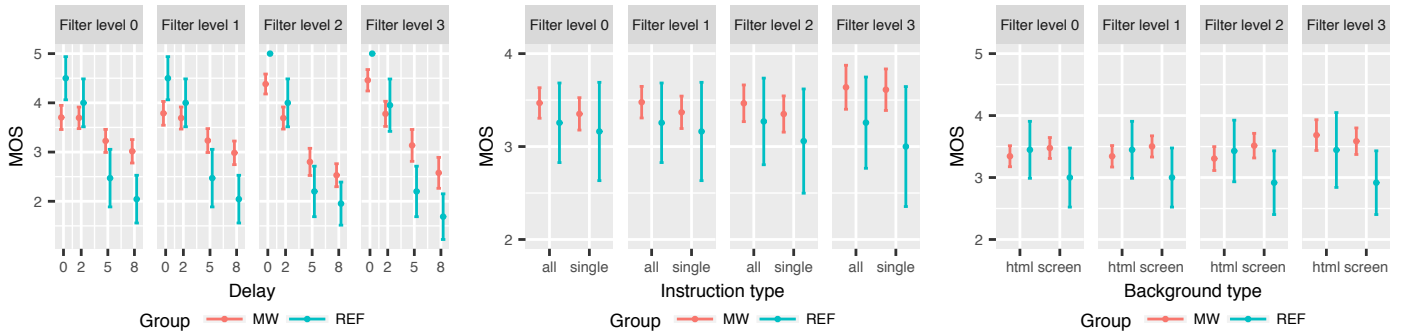
In order to demonstrate the capabilities of our implementation we generate a mock-up of the rather complex Amazon.com webpage to evaluate the impact of page loading delays on the users' QoE. We chose Amazon as an example of a complex web site to demonstrate the simple applicability of our methodology by non-web experts. In the test, users shall purchase a specific scooter while the different process steps, *search*, *selection*, *add to cart*, and *checkout* are delayed. To compare our methodology to traditional means of web-QoE testing, we generate an HTML-based (traditional) and a screen-shot based (our method) emulation of the webpage.

A. Workflow and Survey Configuration

Pre-processing Step: We generate an HTML-based and a screen-shot based emulation of the Amazon.com webpage. For providing the HTML-based approach we download the specific webpages for each step. For the screenshot-based approach, the corresponding web pages are captured using http://web-capture.net/.

Editor: For both approaches we assemble the workflow based on the captured webpages as presented in Section IV. HTML code or screenshots are uploaded in the editor and the specific process steps are dropped to the index tabs. Invisible input fields and buttons are placed over the relevant elements of the webpage templates and the corresponding actions are assigned. One the initial page we use an input field placed on the search field matching for the keyword "Scooter"'. After matching the keyword, the selection page is displayed. Here, an invisible button is placed via an specific scooter linking to the dedicated product side. Invisible buttons placed over the "add to cart" button and the "checkout button" on the following pages finalize the desired workflow. We further add detailed instructions explaining each single step of the workflow. For both approaches, we generate an emulation environment showing all instructions per step and one showing only the specific instructions per step. The corresponding workflows are downloaded as HTML file and used for the user survey. Despite the complexity of the Amazon website, it took about 25 minutes to prepare a test case, independent of the used approach, which demonstrates the simplicity of our methodology.

Deployment and Survey: The captured webpages, i.e., the screenshots or the HTML code, are placed on a webserver, together with the corresponding HTML file for the specific workflow. Further, a jQuery bundle is added to the folder. The workflow is then embedded in a user survey consisting of demographic questions and questions about the participants



(a) Impact of delay on user ratings (b) Impact of instructions on user ratings (c) Impact of implementation approach on user ratings

Fig. 2. Main results of the conducted user study. The users acquired via commercial crowdsourcing are marked as MW, the voluntary participants as REF.

Internet usage behavior. The results are both used to identify additional influence factors on the rating behavior of the users, but also to enable consistency test of the participants to estimate the reliability of the ratings [25].

We use the commercial crowdsourcing platform Microworkers.com in order to acquire a large number of test participants (MW). Microworkers.com focuses on small-granular crowdsourcing tasks that can be completed within a few minutes and are usually rewarded with a few US cent and provides an international user base [26]. For the following evaluation we analyse the results from 512 participants acquired via the crowdsourcing platform. The test was conduced in two phases, the first between 17. Feb 2016 to 18. Feb. 2016 the second between 11. Mar. 2016 and 12. Mar. 2016. Each participant was rewarded 0.10 USD for a response. In parallel, we performed the same test with a anonymous reference group (REF) consisting of 80 volunteers acquired via Facebook and university students.

B. Reliability of Participants

Both the crowdsourcing users as well as the reference group member were completely anonymous and performed the test in an unsupervised environment. This imposes the need for reliability checks in order to filter unreliable participants that submit random ratings. In accordance with the best practices developed in [25], we apply different filters that are not based on the subjective ratings but on disjoint objective measures. Therefore we introduce the following filter levels for the later evaluation:

Filter level 0: All responses are considered.

Filter level 1: This filter users the answers given in the survey before and after the actual QoE test to filter participants. Each participant is requested to fill in his home country in the first survey and the continent he is living on in the second survey. If both information do not match, it can be assumed that the participant did not answer truthfully or clicked randomly, thus his rating it no considered. Further, after completing the test each user is asked to select the product was supposed to buy from a drop-down menu. Every user selecting another option then "scooter" is also removed.

Filter level 2: Here, the same rules as in filter level 1 are applied and an additional plausibility test is performed. The second survey included a question checking, if the participant noticed a delay. This questions can be objectively answered if no delay was induced in the test. So we filter all users who claimed to notice a delay in this case. Further, a delay should obviously be noticed for a emulated delay of 5 and 8 sec. Here, we removed users who were not aware of a delay in these settings. For the test setting using 2 sec, the delay experience cannot be evaluated objectively. Thus, users judging this setting were not considered in this filter here.

Filter level 3: This filter is based on the rules of filter level 3 and an additional evaluation of the completion time of the tasks. Several tests with non-anonymous volunteers showed that the emulated shopping task without additional delay can be completed within 40 sec. We assume that the test settings with additional delay take about the same time to complete, with the additional duration of the artificially introduced delays. Therefore, we discard all ratings of users who worked longer than 180 sec on the shopping task, after subtracting the artificial delays. Participants taking more them that this threshold are likely to perform additional tasks besides the emulated shopping task. Consequently, they might not experience the delays of the page load time and thus are removed from the evaluation.

Impact of Filters on the Number of Participants: The impact of the different filter on the crowdsourcing participants and the reference group are illustrated in Table I. For both groups, filter 1 has a negligible impact on the overall number of participants. Filter 2, however, results in a significant reduction of participants. After applying filter 3, the duration-based filter criteria, less than half of the participants remain in the crowdsourcing group, and about 78% in the reference group.

	f0	f1	f2	f3
MW	512	490	354	237
REF	80	80	71	63

TABLE I. USERS REMAINING AFTER APPLYING DIFFERENT FILTERS

C. Evaluation

In the following we discuss the results of different factors on the overall user ratings. This includes the impact of different constant delays on the user perception, screenshots or HTML-based emulation, and how instructions are displayed. The results are depicted in Figure 2.

Impact of Delay: Firstly, we highlight the impact of the delay on the user rating. The results are highlighted in Figure 2(a). The x-axis of the single plots depicts the perceived delay, the y-axis the MOS. For the microworkers evaluations, it can be seen that increasing delay results in lower user ratings. For filter level 0 and 1 this behavior is not very significant,

but with increasing filter level this can be followed better. For the reference group, the relationship between delay and user rating is clearly visible for all filter levels. Again, for higher levels, it gets extremer with either good or bad ratings, i.e., intermediate ratings disappear. Differences in the ratings between both groups may be due to additional factors like cultural background or varying delay expectations.

Impact of Instructions: Secondly, we focus on the impact of the placement of instructions on the user ratings. For that we compare the ratings in case of all instructions being visible during the delay emulations and only the specific instruction for the current task being visible. The results are illustrated in Figure 2(b). It can be seen that, independent of the specific filter level, the user ratings are very similar for both parameters. Hence, we can conclude that the type of instructions does not have an impact for small workflows.

Webpage vs. Screenshots: Thirdly, we investigate the impact of the chosen approach for generating the delay emulation. Results are shown in Figure 2(c). For all filter level and user group combinations it is not possible to proof a different user-rating for screenshot and HTML-based approach. For this experiment, we can conclude that using our screenshot based approach for generating the test case results in similar user ratings as compared to the traditional and more complex, HTML-based approach. This shows that our approach does not introduce a bias as compared to traditional means of testing. Thus, it can be used for a simpler test design using the screenshot based approach without a significant impact on the outcome of the survey.

VI. Conclusion

To make the assessment of delay effects accessible to a broader audience and to be applicable to a broader set of applications, this paper introduced a generalized testing methodology and realization as open-source tool. It enables the reproducible investigation of delay effects in arbitrary workflows for a general class of systems, including web based systems and non-web based systems such as SAP enterprise solutions. By exploiting screenshots to represent the system under test overlayed with invisible user interface elements composed in a visual editor, it can represent an arbitrary class of systems as web applications. It therefore also enables testing a broad range of non-web systems (e.g., enterprise systems) as web app, e.g., to allow crowdsourced tests that would otherwise not be feasible. Is is further designed such that a broad range of scientists is able to use it without any additional domain knowledge. It features direct deployment of the generated test cases on web-servers and crowdsourcing platforms and the persistent storage of test conditions and configurations in a simple way. We applied our methodology to web QoE as widely used use case to show its general applicability. The obtained results of the performed crowdsourcing study highlight that our methodology does not bias the results as compared to traditional methods for web QoE assessment.

Acknowledgments

The authors express their gratitude to Andre Rentsch for the stimulating discussions during the course of this work. This work is supported by the Deutsche Forschungsgemeinschaft (DFG) under Grants HO TR 257/41-1 and by kubus IT. The authors alone are responsible for the content.

References

[1] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Tutorial: Waiting Times in Quality of Experience for Web based Services," in *IEEE QoMEX*, 2012.

[2] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, "Time is bandwidth"? narrowing the gap between subjective time perception and Quality of Experience," in *IEEE ICC*, 2012.

[3] D. Strohmeier, S. Jumisko-Pyykko, and A. Raake, "Toward task-dependent evaluation of web-QoE: Free exploration vs. "who ate what?"," in *IEEE Globecom Workshops*, 2012.

[4] D. Strohmeier, M. Mikkola, and A. Raake, "The importance of task completion times for modeling web-QoE of consecutive web page requests," in *IEEE QoMEX*, 2013.

[5] I. Arapakis, X. Bai, and B. B. Cambazoglu, "Impact of response latency on user behavior in web search," in *ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.

[6] M. S. Jake D. Brutlag, Hilary Hutchinson, "User preference and search engine latency," in *JSM Proceedings, Qualtiy and Productivity Research Section*, 2008.

[7] S. Stefanov, "Yslow 2.0," in *China Software Developers Network*, 2008.

[8] G. Linden, "Make data useful," 2006.

[9] E. Schurman and J. Brutlag, "Performance related changes and their user impact," in *Velocity – Web Performance and Operations Conference*, 2009, http://www.youtube.com/watch?v=bQSE51-gr2s.

[10] J. Brutlag, "Speed matters for Google web search," 2009.

[11] T. Hoßfeld and M. Fiedler, "The unexpected qoe killer: When the network emulator misshapes traffic and qoe," in *IEEE QoMEX*, 2015.

[12] D. Guse, S. Schuck, O. Hohlfeld, A. Raake, and S. Möller, "Subjective quality of webpage loading: The impact of delayed and missing elements on quality ratings and task completion time," in *IEEE QoMEX*, 2015.

[13] H. Cui and E. Biersack, "Trouble shooting interactive web sessions in a home environment," in *ACM SIGCOMM Workshop on Home Networks*, 2011.

[14] J. Nielsen, "Usability engineering," 1993.

[15] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 1968, pp. 267–277.

[16] B. Shneiderman, "Response time and display rate in human performance with computers," *ACM Computing Surveys (CSUR)*, vol. 16, no. 3, pp. 265–285, 1984.

[17] D. F. Galletta, R. Henry, S. McCoy, and P. Polak, "Web site delays: How tolerant are users?" *Journal of the Association for Information Systems*, vol. 5, no. 1, 2004.

[18] International Telecommunications Union, "G.1030: Estimating end-to-end performance in IP networks for data applications (02/2014)," 2014. [Online]. Available: http://digitus.itk.ppke.hu/~gosztony/ITU-T%20Ajanlasok/E.800-0809_QoS-NP-Defin.pdf

[19] D. Guse, S. Egger, A. Raake, and S. Möller, "Web-QoE under real-world distractions: Two test cases," in *IEEE QoMEX*, 2014.

[20] A. Sackl, P. Casas, R. Schatz, J. Lucjan, and R. Irmer, "Quantifying the impact of network bandwidth fluctuations and outages on web QoE," in *IEEE QoMEX*, 2015.

[21] F. F.-H. Nah, "A study on tolerable waiting time: how long are web users willing to wait?" *Behaviour & Information Technology*, vol. 23, no. 3, pp. 153–163, 2004. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01449290410001669914

[22] G. Rose, H. Khoo, and D. W. Straub, "Current technological impediments to business-to-consumer electronic commerce," *Communications of the AIS*, vol. 1, no. 5es, p. 1, 1999.

[23] B. D. Weinberg, "Don't keep your internet customers waiting too long at the (virtual) front door," *Journal of Interactive Marketing*, vol. 14, no. 1, pp. 30–39, 2000.

[24] B. HernáNdez, J. JiméNez, and M. J. Martín, "Key website factors in e-business strategy," *International Journal of information management*, vol. 29, no. 5, pp. 362–371, 2009.

[25] T. Hossfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Best Practices for QoE Crowdtesting: QoE Assessment with Crowdsourcing," *Transactions on Multimedia*, vol. 16, Feb. 2014.

[26] M. Hirth, T. Hoßfeld, and P. Tran-Gia, "Anatomy of a crowdsourcing platform-using the example of microworkers. com," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*. IEEE, 2011, pp. 322–329.