CrowdStudy: General Toolkit for Crowdsourced Evaluation of Web Interfaces

Michael Nebeling, Maximilian Speicher and Moira C. Norrie
Institute of Information Systems, ETH Zurich
CH-8092 Zurich, Switzerland
{nebeling,norrie}@inf.ethz.ch, maximilianspeicher@gmx.de

ABSTRACT
While traditional usability testing methods can be both time consuming and expensive, tools for automated usability evaluation tend to oversimplify the problem by limiting themselves to supporting only certain evaluation criteria, settings, tasks and scenarios. We present CrowdStudy, a general web toolkit that combines support for automated usability testing with crowdsourcing to facilitate large-scale online user testing. CrowdStudy is based on existing crowdsourcing techniques for recruiting workers and guiding them through complex tasks, but implements mechanisms specifically designed for usability studies, allowing testers to control user sampling and conduct evaluations for particular contexts of use. Our toolkit provides support for context-aware data collection and analysis based on an extensible set of metrics, as well as tools for managing, reviewing and analysing any collected data. The paper demonstrates several useful features of CrowdStudy for two different scenarios, and discusses the benefits and tradeoffs of using crowdsourced evaluation.

Author Keywords
Web usability; user testing; crowdsourced evaluation

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Evaluation/methodology*

INTRODUCTION

Usability evaluation is an important part of the user interface design process. Particular attention has been devoted to web usability, where specific design methods [19] and evaluation metrics [10] have been crafted over the years. While the most commonly used evaluation method is user testing [9], it is heavily constrained by available time, money and human resources. At the same time, an evolving set of tools for automated usability testing have emerged over the years [11]. However, there are still several limitations of current usability evaluation tools that we aim to address in this work.

EICS'13, June 24–27, 2013, London, United Kingdom.

First, given today's proliferation of web-enabled devices, the usability of a web site is largely determined by its ability to adapt to the specific device in use. Existing guidelines such as WCAG, the Web Content Accessibility Guidelines[1] by W3C, consist of a set of recommendations on making content accessible with the focus on meeting special needs of users. The W3C web site lists different evaluation and issue reporting tools, but none of them consider the rapidly evolving range of use contexts. In particular, there is a need for tools that also support mobile settings and make use of the rich input sensing techniques available on modern touch devices.

Second, while there are advanced frameworks for user activity tracking [1], available implementations such as WebQuilt [7] and Web Usability Probe [3] are often restricted to event logging and usually support this on either the server or client side. Therefore, information is generally collected at a very low level semantically which means that post processing and special log analysis tools are required to visualise and make sense of the collected data.

Third, essential tasks such as subject recruitment including qualification tests and many key aspects of remote usability testing including task distribution within or between subjects are generally out of scope of existing tools. Rather than allowing tools to be configured for different settings, often special solutions need to be developed for testing under the required conditions.

Recently, crowdsourcing services such as Amazon Mechanical Turk[2] have received a lot of attention as a platform for conducting online user tests [12]. Research has started to investigate the benefits and tradeoffs of crowdsourced user testing by comparing lab and remote studies [4, 16]. Our goal is to tightly integrate support for crowdsourcing into usability testing tools in order to support a wide range of evaluation methods and scenarios.

In this paper, we present CrowdStudy, a general framework and comprehensive web site testing toolkit that integrates with crowdsourcing services such as Mechanical Turk to advertise and facilitate online evaluations. CrowdStudy evolved out of several research projects that required user testing for a wide range of use contexts in a short amount of time [22, 23]. While it was first specifically developed to conduct these studies [21], we have now built several mechanisms

[1] http://www.w3.org/TR/WCAG20
[2] http://mturk.com

into CrowdStudy that allow it to be configured for different tasks and additional metrics required for particular evaluation scenarios. As examples of important scenarios that pose interesting challenges, we report on two evaluations conducted with CrowdStudy, where it was used for usability testing in large-display environments and on mobile touch devices.

The next section discusses related work. This is followed with two scenarios that are difficult to address with existing tool support. We then present the CrowdStudy framework, its architecture and implementation, together with example studies illustrating its use. Finally, the paper analyses the contributions made with CrowdStudy, compares traditional usability testing and crowdsourced evaluation, and discusses the benefits and trade-offs of each technique.

BACKGROUND

CrowdStudy builds on existing frameworks for automatic usability evaluation [1, 3, 7]. Common to most approaches is logging of user interface events with the goal of extracting usability-related information [5]. While a comprehensive overview is given in [11], here we limit our review to a selection of tools and highlight differences to CrowdStudy.

An early tool is WebQuilt [7] which supports logging and user activity tracking based on server-side components using a proxy-based solution to intercept the interaction with a web site. WebQuilt records the communication between client and server in terms of navigation and access paths within and between web pages. This information can then be visualised in a graph showing web pages as nodes and actions as edges. Given this information, it is possible to detect certain user interaction patterns as well as issues with the document structure and navigation. Compared to client-based solutions, one of the greatest disadvantages of WebQuilt is that recording and handling of JavaScript-based actions is not supported. This is especially problematic given the increased popularity of libraries such as jQuery[3] and that many modern web sites make extensive use of AJAX for dynamic content.

The general framework developed in [1] also builds on a proxy server so that no manual modification of the web site under investigation is necessary. In contrast to WebQuilt, the approach focuses on JavaScript-based, client-side interaction tracking techniques. This is motivated by the fact that such client-based techniques tend to provide richer information in terms of the interaction within a web page. The data that can be collected with that framework includes mouse movements and clicks, element focus and selection, form input and the required time for different form fields. It can then be mapped to the respective page elements and may be used to visualise the interaction paths of users within a web page.

More recently, Web Usability Probe (WUP) [3] was proposed. WUP extends the principles described above in that it not only supports data logging and visualisation, but also automatic analysis. The approach is based on "optimal" logs defined by the evaluator for the test scenario. These logs then provide a reference for comparisons with the logs produced by participants. Similar to other solutions, WUP enables evaluations across web sites while capturing most of the standard mouse and keyboard events. In addition, also custom client-side events can be registered for tracking, giving more flexibility to evaluators. However, visualisation of the recorded data is limited to timelines, which is only useful for time-related performance measurements.

The aforementioned solutions provided good starting points for our framework. However, they lack support for context-awareness which is a core component of CrowdStudy. Specifically, we show how to make use of state-of-the-art sensing techniques [6] to obtain more information on the use context, which is particularly important in mobile settings. In addition, our solution integrates support for crowdsourcing, which has only recently been considered for usability testing [16].

Crowdsourcing refers to the idea of outsourcing a task to a larger group of people in the form of an open call [8]. As already mentioned, much attention has been devoted to paid micro-task crowdsourcing markets such as Amazon Mechanical Turk (MTurk). First studies have assessed it as a general platform for conducting online user studies [12], some of which replicated previous lab studies and obtained similar results [4, 13].

While other methods for usability evaluation might produce more reliable results, MTurk is commonly regarded a useful tool and arguably has advantages over traditional lab experiments, such as easy and quick access to a large user pool, relatively low cost, and faster iteration between initial and follow-up experiments to refine the evaluation procedure [18]. To address the shortcomings of MTurk as a platform for experiments, many different toolkits have been developed on top of it, ranging from TurKit [15] for programming iterative and parallel crowdsourcing task designs, over Turkomatic [14] for using crowds to do the "programming" of tasks, to AutoMan [2] for fully automatic crowd programming.

CrowdStudy is similar to these works in that it also aims to leverage existing crowdsourcing services such as MTurk to facilitate large-scale web site user testing under time and monetary constraints. However, in contrast to these toolkits, CrowdStudy was specifically designed to provide flexible support for crowdsourced evaluation of existing web sites independent of other services, using MTurk as an additional, but optional, channel for conducting online experiments.

The closest to CrowdStudy in terms of the overall design is TurkServer [17]. TurkServer aims to be a general platform for synchronous and longitudinal online experiments, addressing common challenges such as the technical setup of online experiments, user and data tracking across experiment sessions, and filtering of incomplete and invalid data. It does this by providing suitable abstractions and infrastructure, which is similar to CrowdStudy. However, CrowdStudy implements tasks and metrics that are specific to web usability evaluation, and also addresses the proliferation of new devices and how they may impact user experience. As a result, CrowdStudy can be configured to target and recruit more users with devices that are poorly supported by the current web design.

[3] http://jquery.com

This helps developers to target usability problems on specific devices and ultimately provide more flexible web interfaces that can adapt to a greater variety of use contexts.

SCENARIOS

To better illustrate the problems and motivate the techniques developed for CrowdStudy, we present two scenarios that pose different requirements and help to explain the roles of the different components defined in our framework.

Scenario 1: Designing Web Pages for Large Screens

A team of HCI researchers have developed a new prototype of a system that provides end-users with tools for customising a web page specifically for larger viewing sizes. For evaluating the tool support and studying many possible layouts in a short amount of time, they decide to conduct a remote user study and hope to recruit a large number of participants. At first, participants are randomly given one out of three possible tasks—one asking them to adapt the web page, another to compare layouts based on aesthetic considerations and a third for reading using a specific layout and answering questions on the text. Participants are encouraged to work on additional tasks, but task distribution depends on the tasks that a participant has already worked on as well as the number of layouts contributed by other participants. Each task starts by showing instructions and finishes with a post-task questionnaire collecting subjective feedback. The researchers need to be able to closely monitor and inspect the results during and after the study.

Scenario 2: Touch Interaction on Mobile Devices

The same team of researchers have also developed a second prototype specifically for mobile browsers using new techniques for touch interaction tracking and adapting web pages based on user performance measures. Again, they need to evaluate their new system with a large number of participants in a short amount of time, but now for a wide range of different mobile devices. For the study, participants will be given 50 small tasks (e.g. clicking a link) that need to be randomised and counterbalanced. To guide users through the study, respective parts of the web page are highlighted and the window scrolled to the required position. After the study, participants are asked to fill in a questionnaire providing feedback in terms of ratings and comments. The researchers want to review and visually inspect the collected data for convenient and fast analysis. Moreover, a follow-up lab study using a similar set of tasks is planned to validate their findings in a more controlled setting.

Requirements

The above scenarios not only differ in terms of the use context (large screen vs. mobile touch), but also regarding the types of tasks (rather complex tasks such as designing & comparing web pages vs. rather simple, mechanical tasks such as clicking links), task distribution (randomised & controlled vs. randomised & counterbalanced) and assignment (within-subjects vs. between-subjects). Based on these scenarios and our experience with conducting usability studies, we have derived the following set of requirements for CrowdStudy:

(R1) **Context-awareness** The system must be able to detect the client context as well as being compatible with all major browsers, both on desktop PCs and mobile touch devices.

(R2) **Easy integration** The system must be easy to integrate into a web site under investigation. Moreover, smaller changes to the web interface (e.g. for annotating page components part of a task) should be supported.

(R3) **Subject recruitment** The system must support easy recruitment of a potentially large number of participants over a short period of time.

(R4) **Simple and complex tasks** The system must enable both simple and complex tasks, e.g. by splitting tasks and automating irrelevant sub-tasks to keep the cognitive load and participant distraction at a minimum.

(R5) **Controlled testing** The system must support different modes of task distribution and assignment, e.g. randomisation and counterbalancing, within or between subjects.

(R6) **Qualification checks** The system must ensure that only participants fulfilling certain requirements (e.g. usage of a specific browser or device) can take part in the study.

(R7) **Pre/post-conditions** The system must support optional pre and post-conditions for the overall study and individual tasks. For example, conditions could be questionnaires asking for demographics and feedback.

(R8) **Different metrics** The system must provide automatic logging of all kinds of available data, including demographics, user feedback and task-related data (such as task completion time, task success rates etc.). These must be made available in aggregated forms through metrics for convenient statistical processing.

(R9) **On-the-fly data inspection** Based on the metrics just described, the system must provide means for easy and convenient inspection of the gathered data during and after the study, both in terms of individual data sets from single participants, and in configurable aggregated ways.

(R10) **Different types of evaluation** The system must enable easy preparation of different study methods, i.e. asynchronous remote studies as well as controlled lab studies.

CROWDSTUDY

In this section, we give a first overview of our framework from a higher level of abstraction. Figure 1 illustrates the main components of CrowdStudy. The design of these components was informed from previous work [1] and different scenarios including those mentioned previously, but the overall framework design was generalised and also extended to provide support for context-awareness and crowdsourcing. On the one hand, the tasks and metrics components allow for the configuration of usability evaluation scenarios and the information to be collected by the framework. Based on this configuration, it is then possible to automatically control subject recruitment, task distribution and other aspects of a running study. On the other hand, the administrative tools and

components for viewing, analysing and visualising user data are typically used during and after a study in order to monitor tests and evaluate the results. Below we discuss the different aspects of web usability testing addressed by specific components of our framework.

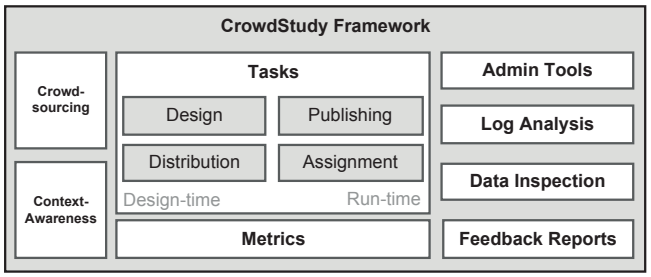

Figure 1: Framework Components

Task Design, Distribution and Assignment *(R4–R5)*
The **Task Design** component allows test developers to define different task designs where each task may involve one or several web site components as well as navigation between pages. Additionally, tasks can be designed for different use contexts and input modalities. Using client-side scripting, some aspects of a task may also be automated to allow users to focus on others. For example, we provide tools that can automatically scroll to a component of interest and annotate and highlight certain parts that require interaction or the attention of users. The **Task Publishing** component refers to the part of the framework that coordinates the invitation of web site visitors to participate in studies. This can also mean to generate task descriptions and publish them on crowdsourcing services such as Mechanical Turk. Finally, the task management components also provide design and run-time support for **Task Distribution** and **Task Assignment**. This means that evaluators can use randomisation or controlled assignment of tasks at design-time and the framework enables automatic counterbalancing at run-time based on the test coverage so far.

Abstract Model for Different Studies *(R6–R7,R10)*
While our framework also enables "traditional" lab studies, it was specifically designed to support asynchronous remote usability evaluation and parallel user tests. This requires special components for task distribution among participants and assignment within or between subjects according to the study design. The task management components of our framework provide flexible support for evaluators to design test scenarios. To support this in a uniform way, we first generalised the basic process of user studies into a simple model that defines the most common steps. We show the abstraction underlying our framework in Figure 2.

Following this model, each step of a study can be introduced by a set of instructions, consists of the user performing one or several tasks and usually ends with a post-task questionnaire before continuing to the next step. Additionally, users may be asked to fill in questionnaires before and after the study to provide information on their background and overall feedback. Since different studies have different requirements, we provide support for varying the basic steps involved. For example, qualification tests are not

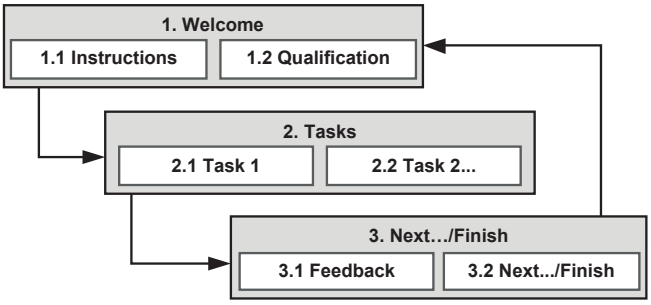

Figure 2: Abstraction of Study Process

required, but depending on the kind of study, they can be implemented either programmatically using technical checks or by means of collecting information explicitly from users. The central entity in this study process is a task defined as $T =< precon, (action)+, postcon >$ where the first and last components refer to the pre and post-conditions, while the actual task is defined in terms of a series of actions the user is asked to perform. Actions may be specifically scripted or can be defined by marking web site elements with pre-defined task classes, e.g., for clicking links, navigating the page or reading text and answering questions. As part of the conditions, tasks can also be configured to require other tasks to be completed first or select the appropriate next task depending on the previous result. This is helpful in the case that they depend on each other and may even be required to enforce a certain workflow in the scope of the investigation.

Extensible Set of Metrics *(R8)*
In addition to this flexible design of studies, our framework supports an extensible set of metrics through the **Metrics** component that can generally be divided into the following classes.

- **Device-related metrics** like the device model and type, display size, resolution and orientation, supported input modalities, browser agent and version, OS, etc.
- **User-related metrics** such as self-reported experience levels in general or specifically with the web site under investigation as well as a user's gender, age, background, skills and preferences
- **Time-based metrics** including time-on-task and intermediate or accumulated times for subsets of tasks or larger parts of a study
- **Counter-based metrics** useful for measuring error and success rates as well as controlled task assignment and distribution, e.g. within and between subjects
- **User activity metrics** including server-side requests and client-side interactions
- **Ratings and comments** for subjective feedback typically collected through questionnaires

Device-related and *user-related* metrics generally describe the use contexts of participants. The device-related ones may

be collected automatically by extracting the relevant information from browser information or using device input sensing techniques based on event listeners. The user-related ones mostly need to be provided by participants and can also be used to define a demographic profile of participants. These may include aspects such as a user's handedness and potential motor impairments to address special needs in addition to a user's cultural background and location. As stated previously, such components have not been an integral part of existing evaluation tools even though they may be used to enrich the information that can be collected based on user activity tracking alone [1, 3, 7].

Time-based and *counter-based* metrics are general indicators as they may be used to measure the time a certain task or aspect of a study required—or to set a time limit for completing a certain task or sets of interrelated tasks—and to then build and compare the error and success rates between participants. Timers and counters may therefore help control the testing environment which is important in remote usability testing. Our framework provides a set of timer and counter components that generate time information and update counters per user session, which can be used to control the time-on-task and task flow. They are usually hidden from users and the control typically remains with the evaluator running the study. The quantitative information collected by these components is typically indicative of user performance and normally used in combination with other metrics as part of a statistical analysis and to help the interpretation of results.

The *user activity* metrics are among the most distinct features of our usability evaluation framework. The framework can be configured to capture client-side events similar to [1, 3]. However, our activity tracking mechanisms operate at a semantically higher level. By this we mean that, rather than simply recording each client-side event together with its type and data in a timeline, we capture interactions that may consist of multiple events and log them on a per-component basis. This allows us to organise the tracking data associated with the components involved in the interaction and also enables our framework to reduce the amount of tracking data. For example, we may aggregate and combine multiple consecutive events, such as scrolling in the same direction, or aggregate individual touch events to higher-level gestures (such as tap-and-pan). In addition, our tools can track, not only click or touch events that successfully activated a link or other kinds of active content, but also those that occurred nearby within a specified range around the target as instances of a potentially intended action that did not get triggered. To support this, we have tools that instrument the page with additional tracking areas surrounding the respective elements. At the same time, our framework automatically registers any changes of the viewport due to scrolling or zooming of the page, changes of the orientation or auto-focus and user-zoom actions. While the latter are primarily supported by mobile browsers and typically performed in response to rotating the device or performing gestures on a touch surface, all of them provide valuable information concerning the use context.

Finally, *user ratings* and *comments* can be collected using single and multiple-choice as well as open questions. However, since the focus of our framework was on other aspects, we only provide basic means for authoring questionnaires, such as JavaScript methods for validating input for required fields. Alternatively, evaluators may combine CrowdStudy with advanced questionnaire tools, e.g. SurveyMonkey[4].

Admin and Data Analysis Tools *(R9)*

As the last set of components, our framework provides **Admin Tools** for creating, managing, testing and deploying new user studies. The analysis tools range from a **Log Analysis** component that allows evaluators to access the data for each test and filter it by certain criteria, e.g. class of device or user experience level, to **Feedback Reports** that summarise questionnaire data provided by participants. In addition, CrowdStudy provides a component for **Data Inspection** of the evaluation data which may be aggregated per task over all participants or per participant for all completed tasks. It is possible to develop different visualisations of the data such as a heat map for touch interaction data aggregated from smartphone or tablet users, as shown in [23].

ARCHITECTURE

Figure 3 shows the typical client/server infrastructure around an existing web application and how the architecture underlying our framework extends it with additional components (marked in grey). Some components of our framework such as the ones for user activity tracking are implemented on the client side, while the context engine and other parts of our framework remain on the server side. The figure also illustrates the integration with existing crowdsourcing platforms such as Mechanical Turk that provide interfaces for programmatic access to their services. Below we describe how each of the components implement parts of our framework and relate to each other.

CrowdStudy Client

First note that clients can either be *users*, i.e. "normal" web site visitors, or *workers* specifically recruited using crowdsourcing services. The components responsible for **User Activity Tracking** are provided by our **CrowdStudy Client** running on the user's device. Depending on the specific metrics and tasks configured for a web site test, CrowdStudy may collect data related to some or all of the metrics described earlier. The data is therefore buffered and cached locally before it is sent to the server side at suitable intervals. Also part of the client-side components is the **Test Pilot** which implements parts of the task management logic for instructing users and guiding them through the tests, e.g. by highlighting and navigating to certain page elements.

CrowdStudy Server

The server-side extensions are grouped into a **CrowdStudy Configuration** and the **CrowdStudy Server**—the heart of CrowdStudy's architecture. The configuration is used for setting up the client-side tool and connecting it to the server.

[4]http://www.surveymonkey.com

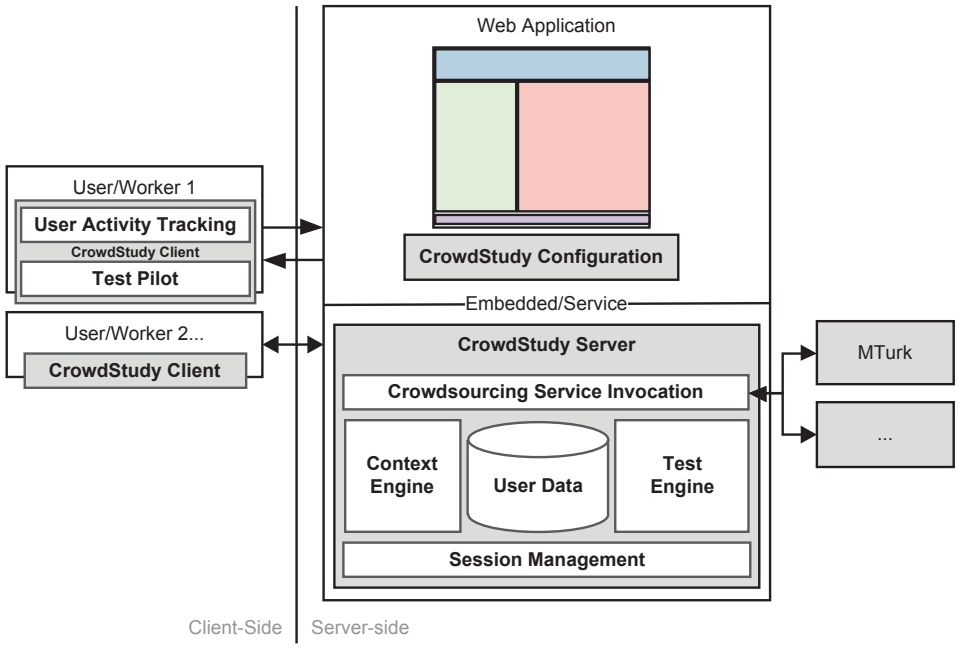

Figure 3: Architecture

The server implements the **Context Engine** that is responsible for associating any data gathered from clients with the corresponding client context. This means that device and user-related metrics from our framework are managed mainly by this component. The server also contains the **Test Engine** which implements the task manager logic responsible for assigning and distributing tasks to participants. All information collected by these components is managed as part of the **User Data** in a database. The **Session Management** component is necessary for identifying and keeping track of users as required for logging. If external, paid crowd workers are to be recruited, the **Crowdsourcing Service Invocation** component enables crowdsourced evaluation via selected platforms.

Integration with Mechanical Turk

Using the MTurk SDK[5] for Amazon Mechanical Turk for example, CrowdStudy can generate and publish so-called *Human Intelligence Tasks* (HIT) on behalf of the evaluator. To generate a HIT, our **MTurk** component accepts several configuration parameters. First, the external URL where CrowdStudy is hosted needs to be specified. Second, a requester needs to decide how long workers can work on a task and how high their reward should be. Each HIT generated by our framework maps all three phases of the study process to MTurk concepts. For the first phase, the qualification ensures that a candidate user is eligible to participate in the study. This can either be done by asking users to answer certain questions, completing sample tasks or based on an assessment using MTurk's own qualification tests setting conditions for HIT completion rate or required a certain demographic profile. In addition, CrowdStudy's context information can be used, e.g. to determine whether the device characteristics match the study requirements. In the second phase, the actual user testing takes place by redirecting workers to a CrowdStudy instance asking them to go through a one or multiple tasks as specified by the evaluator. Finally, the third phase typically includes a feedback questionnaire to gather data about the user's background and experience as well as self-reported measures. CrowdStudy then redirects back to MTurk to allow participants to check the data collected during the study and submit it for review to the evaluator.

Figure 4 illustrates the interaction for a generated HIT from the worker perspective. The management of the HIT is done through Amazon's web site which provides the worker with interfaces for accepting, returning and submitting HITs. To the evaluator, Amazon provides management tools to accept or reject the HITs submitted by the workers (not shown here). The *User Study* and *Questionnaire* parts of the study are hosted on the CrowdStudy Server which collects all measures along the defined metrics including user feedback. The two back-ends are synchronised by consistently using MTurk's assignment identifiers which uniquely describe a (HIT, worker) pair. Currently, evaluators need to manually accept or reject HITs after analysing the data users have submitted to the CrowdStudy Server. Automatic validation could be implemented based on MTurk's API to aid scalability.

Deployment

While most existing frameworks are proxy-based, our solution can either be directly embedded into web sites by the providers or be made available to users in the form of a browser plug-in. The first embedded deployment only requires linking the CrowdStudy Client to the web site using a single line of code similar to including JavaScript libraries such as jQuery. The latter service deployment is similar to Mozilla's Test Pilot[6] which is a plug-in for collecting struc-

[5]http://aws.amazon.com/mturk

[6]http://testpilot.mozillalabs.com/

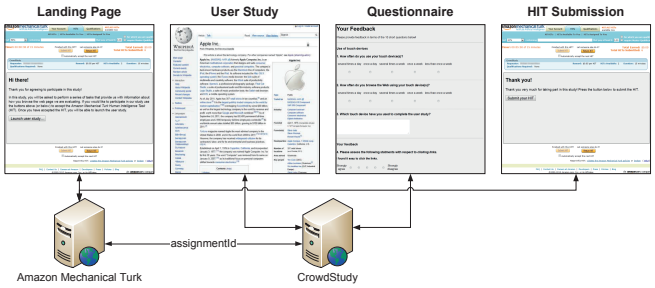

Figure 4: Integration with Amazon Mechanical Turk

tured user feedback through Firefox. Using the CrowdStudy plug-in, other developers could create and exchange test scenarios for an existing web site independent of the provider.

The embedded deployment mode can be useful for focus tests of new parts of a web site before they are rolled out to the entire user community. The service deployment mode is required for platforms such as WordPress and Facebook, where third-party application developers may want to test how their solutions integrate with the kernel application and platform, but have no control over the hosting site.

IMPLEMENTATION AND USE

CrowdStudy is a lightweight framework and does not impact web site performance. To achieve this, it uses asynchronous synchronisation with the server and implements client-side buffering mechanisms not to interfere with the collection of timing data during active tasks. The CrowdStudy Client is implemented using jQuery in combination with jQMultiTouch [20] for handling touch events and device orientation on mobile devices. The CrowdStudy Server is implemented in PHP, using MySQL as the database backend.

CrowdStudy can be added to an existing web site simply by embedding the CrowdStudy Client as external JavaScript and providing a configuration. CrowdStudy works best on web sites implemented in HTML/CSS, where all elements involved in tasks are accessible via the DOM. Notable exceptions are Flash, Silverlight and custom HTML5 canvas implementations, where interaction tracking is limited to the container element that renders the inner-content. The highly dynamic nature of modern web sites is generally not a problem for CrowdStudy's interaction tracking, but may require additional techniques for conditional data inspection and visualisation. Next to recording user interactions, CrowdStudy can also collect qualitative data in the form of questionnaires.

All data collected by the CrowdStudy Client is associated with OS and browser information, screen dimensions and display orientation. In addition, CrowdStudy uses MobileESP[7] for detecting mobile device classes and further distinguishes portrait and landscape mode using a combination of JavaScript and CSS3 media queries. For the integration with MTurk, CrowdStudy generates HITs using an *external question* to point to the CrowdStudy URL that MTurk embeds by using an HTML iframe of configurable size.

[7]http://mobileesp.com

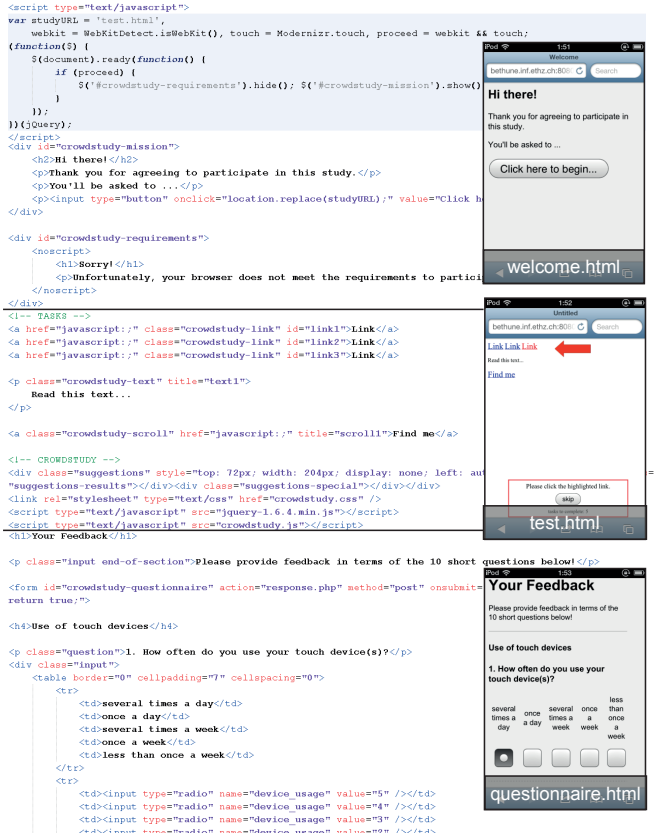

Figure 5: CrowdStudy Example

CrowdStudy offers several pre-defined tasks and metrics. Web page elements part of a task only need to be annotated with special CSS marker classes. New types of tasks can be implemented using jQuery callbacks. Likewise, additional metrics can be registered by implementing the relevant tracking functions in the form of callback handlers on the client and/or server side as required.

Figure 5 illustrates the steps required for setting up a CrowdStudy instance for a simple example page. Similar to the second scenario, crowdsourced evaluation is used in a mobile context. The **welcome.html** page implements a simple browser and device check using JavaScript and welcomes participants or shows a corresponding error message. The **test.html** page defines several tasks by associating links and text paragraphs in the page with pre-defined CrowdStudy task classes. By linking jQuery and **crowdstudy.js**, the CrowdStudy Client will automatically analyse the DOM and compile a set of tasks accordingly. The **questionnaire.html** page collects user feedback once all tasks are completed.

Interested readers are referred to the the project web site[8] for more technical information and the CrowdStudy source code.

EXAMPLE STUDIES

This section presents two experiments that we conducted using CrowdStudy based on the scenarios described earlier. We focus on these examples because they demonstrate different

[8]http://dev.globis.ethz.ch/crowdstudy

aspects of CrowdStudy due to different study requirements. The first experiment illustrates how CrowdStudy can be configured for task distribution and assignment according to the requirements and different phases of a study and dependencies between tasks. The second study shows examples of how CrowdStudy was extended with new metrics specifically designed for multi-touch interaction and additional visualisation techniques for touch input tracking data. The tasks used in the experiments are shown in Figure 6 and the metrics in Figure 7. Note that the two studies are not a contribution of this paper, but the fact that CrowdStudy supported them is meant to demonstrate its usefulness and that it can be configured for different scenarios. Rather, this section contributes an analysis and comparison of the two experiments in terms of the tasks and metrics implemented using CrowdStudy. Interested readers are referred to [22, 23] for details on these studies.

Study 1: Designing Web Pages for Large Screens

The first study concerned the quality of new web page layouts created by end-users as well as the set of new tools proposed for adapting web sites to particular viewing conditions. The experiment consisted of three tasks illustrated in Figure 6 (left) and was divided into two phases.

The *design* task asked participants to use the design tools to adapt the layout of a news article so that it best supported their viewing situation. The *compare* task asked participants to rate a number of layouts for the news web page by comparing two layouts in each step and choosing the better one for reading the article. The *evaluate* task asked participants to read the news article using one of the layouts and then answer five questions on the text by clicking on the text paragraph that contained the answer, rather than typing it.

For each task, CrowdStudy showed instructions and measured the time for completion. In the *design* task, CrowdStudy was also used for logging how users made use of the design tools as well as when the window was resized. In the *compare* task, participants were not allowed to vote on their own layout, and CrowdStudy only counted one rating per pair for each participant even though it was possible that the same pair was compared multiple times. In the *evaluate* task, the times for reading and answering were measured separately. The *design* and *evaluate* tasks finished with a post-task questionnaire, where CrowdStudy was used for data collection.

CrowdStudy also controlled task distribution and assignment within and between subjects. The experiment initially started with the original design of the news article used as the basis for designing and reading. This was switched in the second phase of the experiment, then using the currently best-matching user-generated layout for the tasks. CrowdStudy randomised between the three tasks, but the *design* and *evaluate* tasks were allowed only once within subjects. The *compare* task was only enabled for client contexts that matched at least three different layouts to reduce the chance that participants would see their own layout in comparisons.

We recruited a total of 93 participants using CrowdStudy. Users had to pass a CrowdStudy qualification page checking whether their browser settings met the technical requirements of the experiment as only Firefox and no mobile devices were supported. Overall, CrowdStudy monitored the design of 28 custom layouts, collected 143 ratings and 42 answers providing reading feedback for several layouts. It also helped us to coordinate switching to the second phase once we had received sufficient user-generated layouts and ratings [22].

Study 2: Touch Interaction on Mobile Devices

The goal of our second study was to conduct a comparative evaluation of the original Wikipedia web site and a new version generated for touchphones and tablets. The study involved both crowdsourced usability evaluation and a smaller follow-up lab study to test the validity of results. Since participants were allowed to use their own touch devices in the remote evaluation, CrowdStudy was tested in terms of compatibility with a wide range of mobile devices and different browsers. Using CrowdStudy, we implemented device and browser checks as part of the qualification test to make sure that users had a touch-enabled device and adequate browser support before they were allowed to participate.

The experiment consisted of four different types of tasks illustrated in Figure 6 (right). In *click link* tasks, participants were asked to tap on a specific link that was underlined and highlighted as well as marked by an arrow pointing at the link. For *find link* tasks, the article was scrolled to the top and participants then had to find a link within the main article text marked by our framework. The time required to locate and click the correct link was measured, as well as the number of times the scrolling direction changed. In *read text* tasks, participants were asked to read a highlighted part of the article text. In this case, the instructions box additionally provided a "Done" button for users to indicate that they finished reading, after which we logged the required time. In *describe image* tasks, users answered a multiple-choice question for which it was necessary to view details of an image. The image was scrolled into the viewport and marked. CrowdStudy counted correct responses and recorded the time required to answer.

In contrast to the first study, all tasks were carried out within the same page. A total of 33 tasks of different types were defined, randomised and counterbalanced between subjects using CrowdStudy. In order that users only had to focus on the actual interaction that was required to complete a task, CrowdStudy automatically scrolled to the respective part of the web page except for the *find link* task that started from the top. For each task, CrowdStudy showed instructions at the bottom of the page and guided users through the experiment.

The first part of the study using crowdsourcing involved 84 participants in total; 39 tested the original version, 45 the adapted one. Here, we used CrowdStudy for closely monitoring the experiment and specifically recruiting smartphone and tablet users to focus testing on mobile settings: 64 participants used a smartphone (we detected a number of different models such as iPhone, HTC Desire, Motorola Defy, Samsung Galaxy S and Nokia N9), the other 20 used a tablet device including iPad, Lenovo ThinkPad, Motorola Xoom and Archos 70. 50 completed the study with questionnaire feedback providing their gender, age and ratings concerning device usage in general and specifically for web browsing. In

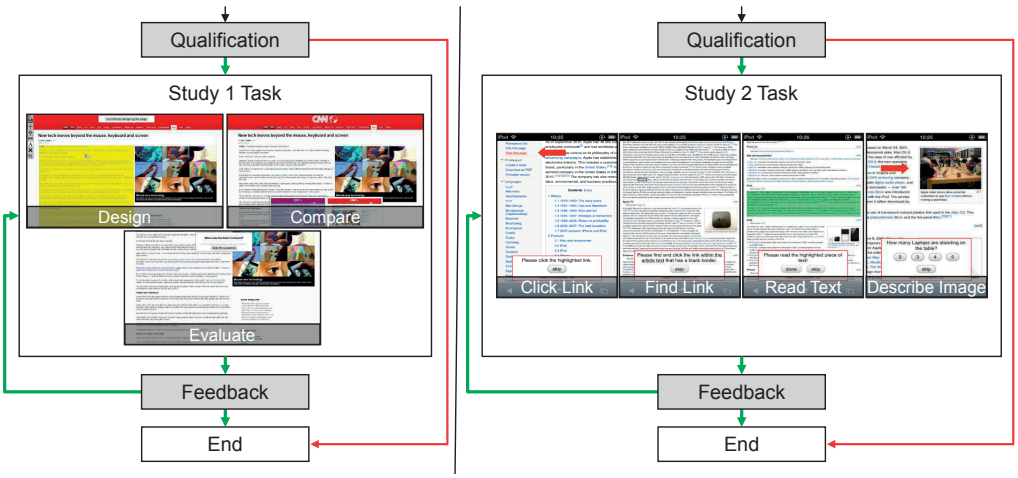

Figure 6: Tasks used in Study 1 (left) and Study 2 (right)

Metrics	Study 1: Designing Web Pages for Large Screens	Study 2: Touch Interaction on Mobile Devices
Device-related	Detected browser and screen resolution; recorded window width and height; collected other system information including language, OS version, and location in terms of the IP address.	Detected mobile operating system, browser and screen resolution as well as device type (using MobileESP); **Per task**: logged the zoom level and device orientation.
User-related	Demographics (gender, age, etc.); also asked for level of expertise in design, development and user experience.	Demographics similar to Study 1; frequency of touch device use in general and for web browsing specifically.
Time and counter-based	**All tasks**: measured completion times for *design*, *compare* and *evaluate* tasks (with separate times for reading and answering); **Compare task**: number of votes per layout and participant; **Evaluate task**: number of questions answered or skipped.	**All tasks**: measured task completion times; counted skipped tasks.
User activity and interaction-based	**Design task**: recorded all customisations in terms of design actions and how web page elements were changed (position, size, spacing, font size, visibility etc.); **Evaluate task**: logged scrolling actions; **All tasks**: logged when and how the window was resized while working on a task.	**Click link task**: measured missed-links and zoom-factor ratios to see how often participants misclicked links and zoomed into certain areas of the page; **Find link task:** additionally logged scroll actions including how often the scroll direction changed; **Read text and describe image tasks**: monitored zoom level.
Ratings and comments	**All tasks**: collected ratings on whether task was easy to understand and easy to perform; question on further comments; **Design task**: also ratings on usefulness of each tool; **Evaluate task**: additional questions on reading efficiency using the layout; **Design and evaluate tasks:** different ratings of layout.	Post-study questionnaire asking participants to rate statements concerning the ease and efficiency of tasks; question on further comments.

Figure 7: Metrics used in the two example studies

the follow-up lab study with 13 participants, a simpler version of CrowdStudy and an iPod touch were used. Based on the collected data obtained in both phases, we developed simple, touch-related usability metrics and visualisation techniques that, not only allowed visual inspection of the collected data by aggregating the results obtained from different users, but also identifying critical components that required adaptation for the particular use context. This is detailed in [23].

DISCUSSION AND CONCLUSION

This paper presented CrowdStudy, a flexible framework for designing and conducting web site evaluations in the lab or remotely, then using crowdsourcing techniques and services such as Mechanical Turk for large-scale user testing. To demonstrate the novelty and flexibility of our framework at the technical level, the studies spanned many different use contexts including different kinds of mobile touch devices.

CrowdStudy is not yet another framework on top of Mechanical Turk (MTurk). Rather, CrowdStudy is a general framework designed for conducting web usability tests that can also be used in combination with crowdsourcing services such as MTurk. While MTurk already provides support for qualification tests, the level of support and implementations vary between different crowdsourcing platforms. CrowdStudy's tests can be similar to MTurk's, but are configured independently of crowdsourcing services and are also available if CrowdStudy only recruits web site users rather than crowd workers. Also note that CrowdStudy tests can go beyond the worker history and their performance, making it possible for usability testers to specifically recruit users and conduct tests for undersupported use contexts. In particular, the two studies show that it is possible to realise dependencies between tasks and control whether and how tasks are assigned to users depending on their background and qualification as well as the device in use and progress of the experiment.

A central question is how crowdsourced evaluation compares with usability evaluation using experts or lab subjects. While this is generally discussed in [18, 16], our experience with using CrowdStudy suggests that the benefits outweigh the trade-offs. For example, the second study conducted with CrowdStudy showed that our mobile test version came very close to the original in terms of reading experience and efficiency,

with similar results in both the online and lab setting. The lab study confirmed the findings of the crowdsourcing experiment, but did not provide any new insights. However, the more controlled setting produced higher validity of results. For example, the reading times showed generally less variance in the lab study. On the other hand, crowdsourced evaluation provided additional insight into how participants used the test page on many different smartphones and tablets under diverse and realistic conditions. This allowed us to detect use patterns and differences, not only between different types of devices, but also between portrait and landscape, which would be difficult in a lab setting.

While CrowdStudy does not provide specific mechanisms for quality control in the form one might expect from crowdsourcing frameworks, it provides tools for reviewing and analysing collected data. In particular, via the admin tools, it is possible to block users and exclude selected contributions by marking them as incomplete or invalid. A possible extension to CrowdStudy is a tool for replaying the user interactions as a basis for qualitative analysis. It could also be interesting to combine CrowdStudy with more advanced crowdsourcing frameworks such as Turkomatic [14] or AutoMan [2]. We plan to extend the framework in mainly two ways. First, we are currently exploring the extension of our framework with additional tracking techniques such as 3D skeletal tracking using Kinect. This will provide additional support for studies with a focus on collaboration that were out of scope at this stage. Second, we aim to support both co-located and remote user studies in multi-device environments by developing new concepts and techniques for interaction tracking across devices. This includes scenarios where the user interface is dynamically distributed and migrated between devices, which requires additional mechanisms.

Acknowledgements

We thank Michael Grossniklaus for his help with the Mechanical Turk integration. This work was supported by the Swiss NSF under research grant 200020_134983.

REFERENCES

1. Atterer, R., Wnuk, M., and Schmidt, A. Knowing the User's Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In *Proc. WWW* (2006).
2. Barowy, D. W., Curtsinger, C., Berger, E. D., and McGregor, A. AutoMan: A Platform for Integrating Human-Based and Digital Computation. In *Proc. OOPSLA* (2012).
3. Carta, T., Paternò, F., and de Santana, V. F. Web Usability Probe: A Tool for Supporting Remote Usability Evaluation of Web Sites. In *Proc. INTERACT* (2011).
4. Heer, J., and Bostock, M. Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design. In *Proc. CHI* (2010).
5. Hilbert, D. M., and Redmiles, D. F. Extracting Usability Information from User Interface Events. *CSUR 32*, 4 (2000).
6. Hinckley, K., Pierce, J. S., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction. In *Proc. UIST* (2000).
7. Hong, J. I., Heer, J., Waterson, S., and Landay, J. A. WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. *TOIS 19*, 3 (2001).
8. Howe, J. The Rise of Crowdsourcing. *Wired 14*, 6 (2006).
9. Insfran, E., and Fernandez, A. A Systematic Review of Usability Evaluation in Web Development. In *Proc. WISE Workshops* (2008).
10. Ivory, M., and Megraw, R. Evolution of Web Site Design Patterns. *TOIS 23*, 4 (2005).
11. Ivory, M. Y., and Hearst, M. A. The State of the Art in Automating Usability Evaluation of User Interfaces. *CSUR 33*, 4 (2001).
12. Kittur, A., Chi, E. H., and Suh, B. Crowdsourcing User Studies With Mechanical Turk. In *Proc. CHI* (2008).
13. Komarov, S., Reinecke, K., and Gajos, K. Z. Crowdsourcing Performance Evaluations of User Interfaces. In *Proc. CHI* (2013).
14. Kulkarni, A. P., Can, M., and Hartmann, B. Collaboratively Crowdsourcing Workflows with Turkomatic. In *Proc. CSCW* (2012).
15. Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. TurKit: human computation algorithms on mechanical turk. In *Proc. UIST* (2010).
16. Liu, D., Lease, M., Kuipers, R., and Bias, R. G. Crowdsourcing for Usability Testing. *CoRR abs/1203.1468* (2012).
17. Mao, A., Chen, Y., Gajos, K., Parkes, D., Procaccia, A., and Zhang, H. TurkServer: Enabling Synchronous and Longitudinal Online Experiments. In *Proc. HCOMP* (2012).
18. Mason, W., and Suri, S. Conducting behavioral research on Amazon's Mechanical Turk. *Behav Res 44*, 1 (2011).
19. Matera, M., Rizzo, F., and Carughi, G. Web Usability: Principles and Evaluation Methods. *Web Engineering* (2006).
20. Nebeling, M., and Norrie, M. C. jQMultiTouch: Lightweight Toolkit and Development Framework for Multi-touch/Multi-device Web Interfaces. In *Proc. EICS* (2012).
21. Nebeling, M., Speicher, M., Grossniklaus, M., and Norrie, M. C. Crowdsourced Web Site Evaluation with CrowdStudy. In *Proc. ICWE Demos* (2012).
22. Nebeling, M., Speicher, M., and Norrie, M. C. CrowdAdapt: Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences. In *Proc. EICS* (2013).
23. Nebeling, M., Speicher, M., and Norrie, M. C. W3Touch: Metrics-based Web Adaptation for Touch. In *Proc. CHI* (2013).