Human Computation (2016) 3:1:99-117

ISSN: 2330-8001, DOI: 10.15346/hc.v3i1.6

CCLTracker Framework: Monitoring user engagement and learning in web-based citizen science projects

JOSE LUIS FERNANDEZ-MARQUEZ, UNIVERSITY OF GENEVA

IOANNIS CHARALAMPIDIS, CERN, SWITZERLAND

OULA ABU-AMSHA, HIGHER INSTITUTE FOR APPLIED SCIENCE AND TECHNOLOGY, SYRIA

S.P. MOHANTY, CERN, SWITZERLAND

ROSY MONDARDINI, CITIZEN CYBERLAB

DANIEL K. SCHNEIDER, UNIVERSITY OF GENEVA, TECFA, SWITZERLAND

FRANCOIS GREY, UNIVERSITY OF GENEVA

BEN SEGAL, CERN, SWITZERLAND

ABSTRACT

Analytics tools have been widely used over the last years for the development of web-based applications and services. Analytics data allows improving user interfaces through planning, executing, and evaluating actions intended to increase user engagement.

Measuring and improving user engagement in citizen science projects is not different from other web applications such as on-line shopping, newspapers, or sites for recommending music or movies. However, many citizen science projects also aim to produce learning outcomes among the participants. Current analytics tools do not present sufficient information regarding user behaviour with the application, thus making measuring engagement and learning outcomes difficult.

This paper presents the CCLTracker analytics framework that is intended to overcome current limitations in analytics tools, by providing an API for monitoring user activities such as time spent watching a video, time to complete a task, or how far down a page is scrolled. CCLTracker has been integrated in two different citizen science projects which have validated its value for measuring user engagement and learning.

1. INTRODUCTION

The explosion of web applications and on-line services over the last 20 years have highlighted the utility of analytics tools for understanding who the users are, how they find the website, and how they behave. Through monitoring key performance indicators (e.g. downloads, sales, or participation on the website), designers can increase user engagement by planning, executing, and evaluating changes to interfaces and messaging.

A web-based citizen science project is no different from other online services in this regard. Dissemination actions are also focused on engaging users and empowering their participation in the project. Citizen science projects require understanding who the users are, where they are coming from, and how they behave in the website. Additionally, many citizen science projects aim to provide learning outcomes. Citizen science projects focus on two kind of indicators: (1) key performance indicators such as number of visitors, number of goals achieved (e.g. number of tasks completed by users), or number of clicks on links to external sources; and (2) key learning indicators such as accuracy of the task done, speed completing the task, % of tutorial and extra contents accessed, or participation in forums or chats. Key performance and learning indicators can be measured using analytics data.

There are many different analytics tools such as Google Analytics(??, Goo), and Piwik (??, Piw; Miller, 2012), which can provide interesting information about the users and the website. These tools store and aggregate the data, and provide a Graphical User Interface (GUI) that allows users without programming skills to manipulate analytics data and create custom reports to periodically monitor key indicators.

However, the information regarding the user behaviour is limited to average time per session, page views, or bounce rate (i.e. percentage of people that leave the site without any interaction). This information is irrelevant for evaluating user participation in citizen science projects, or to understand who the most valuable contributors are. Even though analytics tools provide information about the location of users, it is not enough to evaluate engagement. For example, we could observe that 500 users are coming from Reddit, but still not know if they participated on the citizen science project or simply visited the site.

Monitoring key performance and learning indicators in a citizen science project requires monitoring user behaviour. Both Google Analytics and Piwik allow sending application-specific events, i.e. events designed to be fired from the website when monitoring a user's specific actions. For example, we can add the number of clicks per session, the number of times a user writes on the forum, the clicks to internal and external links, whether the user is scrolling down the page or how much time a user has been watching a video. All these examples require programming a tracking system in the client-side and connecting it with the analytics tool. This tracking system is tough to implement and keep updated over the different versions of the site. Additionally, it creates strong dependencies on the analytics tool chosen.

This paper presents the CCLTracker analytics framework to ease monitoring user engagement and learning in citizen science projects, and overcome current limitations with analytics tool while reducing the implementation effort. CCLTracker's main features are:

– High-level API to ease the creation and manipulation of events. This API reduces the programming effort to monitor the user behaviour.
– Client-side aggregation of data before sending it to the analytics tool. This allows to cross user behaviour data with other data coming from the analytics tools such as demographics or referrals. E.g. we could see how many people have been processing more than 100 tasks in UK, or the gender of people scrolling down the webpage until the end)
– Decoupling between the tracking system and the analytics tools, i.e. an event created from CCLTracker library can be sent to Google Analytics, Piwik or whatever analytics tool is able to capture JavaScript events.

As a proof of concept this paper presents the CCLTracker analytics framework used in two case studies and shows how the framework overcomes limitations of existing analytics tools, namely Google Analytics.

Section 2 presents background information about Google Analytics, their limitations, and learning analytics. Section 3 shows the CCLTracker JavaScript library in detail. Section 4 briefly describes the CCLTracker framework. Two case studies are presented in Section 5 as a proof of concept. Finally, we present conclusions and future work.

2. BACKGROUND

The most popular web analytics tools is Google Analytics (GA)(??, Goo) used by more than 80% of websites using analytics tools, which represents 53% of all websites (??, ana). GA is a free (limited by quotes) online service provided by Google. The information provided is organised in the following three categories:

Audience provides information about who the users are as well as general information about the site performance such as number of users, number of sessions, page views, or average pages per session. Regarding the users, we can gather information about their age, gender, location, interest (affinity category and in-market segment), browser, mobile phone, operating system, screen resolution, and language.

Acquisition provides information about how the users reach the website, i.e. using the URL (direct), from an organic search, a referral or social media. This information helps to evaluate outreach activities.

Behaviour provides information about what the user is doing on the website, e.g. pages visited, or behavior flow, and information about the site performance, e.g. average page load time, or average server connection time. Additionally, this section provides information about user events.

Events usually track user activities within the website, but each of the events requires to be implemented on the website and sent to Google Analytics. For example, the event "task done" would be generated when a user completes a task. Events are extremely important for understanding what a user is doing in the website, measuring engagement, learning, and evaluating outreach activities. Almost any interaction such as internal/external clicks, time spent between clicks, mouse trajectories, or time spent watching a video can be tracked. However, the implementation of the event logic can be complex, requiring a substantial effort to implement it, and making it not affordable for most of citizen science project.

The CCLTracker Framework aims to mitigate this effort by providing a high-level API to ease the creation and manipulation of events, i.e. to easily implement the needed logic to monitor user actions on the website.

There are three major ways of manipulating data in Google Analytics: **Crossing, Segmenting and Filtering**. Google Analytics allows crosschecking data by adding a second dimension, e.g. we can see number of users per country, or number of session coming from Facebook. Segmentation allows us to focus the analytics data on a subset of sessions or users, e.g. users coming from Spain, new users, female users, users between 18 and 24 years old, users coming from Reddit. Filtering allows us to clean the data and filter those that are not relevant. E.g. we could ask for all referrals ordered by average time per session; however, we can find at the top referrals with few users that are not relevant for the stats. In this case we can apply a filter to remove from the list those referrals with less than a given number of users.

Combining information coming from the analytics tool such as gender, location, interest, or referral with user activities is crucial for evaluating engagement, learning and outreach activities. For example we could know if people coming from reddit are participating on the web or just visiting it., or we could know the average age of users participating on the forum, or the age and gender of people contributing every day. However, manipulating personal data such as age, or gender with user actions can be used to find out the age or the gender of one user. Google Analytics prevents those malicious behaviours by applying a threshold, i.e. it does not allow to retrieve information at the same time that allows us to find out user personal information. This threshold prevents us to segment the data base on user actions (e.g number of tasks done, or time participating on the website), and cross it with demographic information (e.g. we can not gather the average age of users contributing in the website for more than 3 hours).

CCLTracker allows us to combine user activities with demographic and all other data provided by Google Analytics while dealing with Google privacy restrictions by using client-side aggregation. Basically, CCLTracker pre-aggregates the data before sending it to the analytics tool, e.g. hours participating on the CS project, or number of tasks done. Since we do not need to aggregate later on the data in Google analytics, we do not have the privacy constraint.

One of the disadvantages of implementing a complex user monitoring system based on Google Analytics is the dependency of the code with Google Analytics. Implementing specific events for monitoring user activities on the website might involve hundreds or thousands of lines of code spread over the project and strongly coupled to Google analytics. If for any reason it is desired to change from Google analytics to another analytics tool, it will require to re-implement most of the code.

CCLTracker JS library is independent of the analytic tool and it allows to switch between analytics tools without requiring major changes on the tracking code.

2.1. Learning analytics

User analytics data are extensively used in the field of business intelligence to support data-driven decision making where the data are in fact the result of tracking the user behavior. Educational online environments have also their own motivations to track learner behavior; these motivations might be pedagogical, administrative or economical (Ferguson, 2012).

A consensus around the definition of learning analytics arose in 2011 at the first International Conference on Learning Analytics and Knowledge (??, LAK). Learning analytics deals with the measurement, collection, analysis and reporting of data about learners and their contexts, for the purposes of understanding and optimizing learning and the environments in which it occurs.

Increasingly, more citizen science projects announce clearly that they have educational objectives in addition to the classical goal of advancing science and research with the help of the crowd. Hence it is natural for an online citizen science project to track participants in order to assess the quality of their participation and the extent of their learning.

Some learning outcomes might be assessed directly from the analytics data, such as the percentage of right answers, or the time of completing tasks. However, this approach is limited since we do not know what the user knew before participating and what the user know after the participation in citizen science projects. Oula Abu-Amsha et al. (Abu-Amsha, Schneider, Fernandez-Marquez, Costa, Fuchs, and Kloetzer, Abu-Amsha et al.) suggested the combination of the CCLTracker framework with surveys, allowing the system to measure learning and correlate that information with the analytics data. Thus, the system can infer which behaviour leads the participants to increase their learning experience. Oula Abu-Amsha et al. (Abu-Amsha, Schneider, Fernandez-Marquez, Costa, Fuchs, and Kloetzer, Abu-Amsha et al.) present a classification of the analytics data that are of interest to assess the learning and the engagement of participants

Engagement is another interesting aspect of the analytics data provided by the CCLTracker framework which has been frequently explored in citizen science research (Ponciano and Brazileiro, 2014; Kloetzer et al., 2017) (How much time a participant stays connected to the website, how often she gets back to it, or during which period of time). Additionally, it is important to know quantitatively how much the participant contributes to the project. H. O'Brien and E. Toms (O'Brien and Toms, 2008) proposed in 2008 a convenient conceptual framework to characterize user engagement with technology-based applications. Combining surveys with the engagement metrics as suggested by Oula Abu-Aisha et al. (Abu-Amsha, Schneider, Fernandez-Marquez, Costa, Fuchs, and Kloetzer, Abu-Amsha et al.) allows to understand how engagement influences user learning outcomes. A deeper analysis of learning outcomes in Citizen Science project is presented in Oula Abu-Amsha et al. (Abu-Amsha, Schneider, Fernandez-Marquez, Costa, Fuchs, and Kloetzer, Abu-Amsha et al.), where the CCLTracker library generates the needed analytics to compute both the learning and engagement indicators.

3. CCLTRACKER JAVASCRIPT LIBRARY

The increasing interest for monitor user activity on websites motivates the extension of analytics tools to provide easy ways of implement it. Google recently lunched Google Tag Manager (GTM). *Google Tag Manager(GTM)* provides a simple graphical interface for defining analytics events (i.e. monitoring user behaviour on the website). These events target some particular DOM elements in the website, identified by some characteristics. Providing this information is a trivial task for simple websites, but is nearly impossible for web applications. That is because there are too many elements to keep track of, and their semantics might even change over time (ex. a 'Start' button might become a 'Stop' button).Therefore we need to have higher-level information that should come directly from the application.

There are already various analytics toolkits available, however they are either overly complicated, or they abstract too much information, making it more complicated for us to process the data[1]. Moreover, most of these solutions are available only with a paid subscription or not available under an open-source licence. In addition, we did not want to lose the core functionality from *Google Analytics* or *Piwik*, so we decided to develop an additive solution.

Therefore, we designed the *CCLTracker Javascript Library* [2] to be a set of tools for simplifying the communication with an analytics provider, while at the same time de-decoupling the work of the developer and the analytics expert. The library works as a proxy between the application and the analytics provider, decoupling the tasks of the developer and the analytics expert. In the following sections we explain in greater detail how the *CCLTracker Javascript Library* works and how it is used in our projects.

3.1. Technical Details

In order to minimise the integration effort, the *CCLTracker Javascript Library* takes care of offloading most of the concerns from the developer. Upon loading, events can be triggered right away using the `analytics.fireEvent(..)` function, without caring about the analytics provider that will receive them. In addition, the library takes care that no events are lost even if the analytics provider becomes available only at a later time.

Internally, the *CCLTracker Javascript Library* keeps a buffer with all the pending events, and waits for a callback from the analytics provider when it is loaded. To be more precise, the library will wait for a fixed amount of time until a function named `analyticsListener` is defined in the global scope. When the `analyticsListener` function is defined, the library will flush all the pending events to the analytics provider and then forward all calls to `analytics.fireEvent`, to `analyticsListener`. Even if this function is never defined, the library assumes that the request was blocked due to tracking/ad-blocking extensions in the browser and pauses the analytics collection. There are two different methods for forwarding the events to the provider:

– Through a **drain function**, that will receive every event being produced, or
– Through **event-binding** using the DOM Window as a proxy element.

The second method is favoured in our case, since the analytics logic is managed by the analytics experts and is delivered through Google Tag Manager. The developer creates a specification document describing all the events triggered by the application, and analysts use GTM to listen for particular events in the window DOM. This way, developers expose all the information required to accurately describe an event, and analysts decide how and what is going to be processed.

This approach is also illustrated in Figure 1. The application triggers events to the *CCLTracker Javascript Library* , which get forwarded to the code provided by the analytics experts through GTM. The big benefit of this approach is the fact that additional events, that were not previously accounted for, can be tracked using the low-level tracking features of GTM or any other tracking provider.

[1] https://segment.com/docs/libraries/analytics.js/
[2] https://github.com/wavesoft/CCLtracker

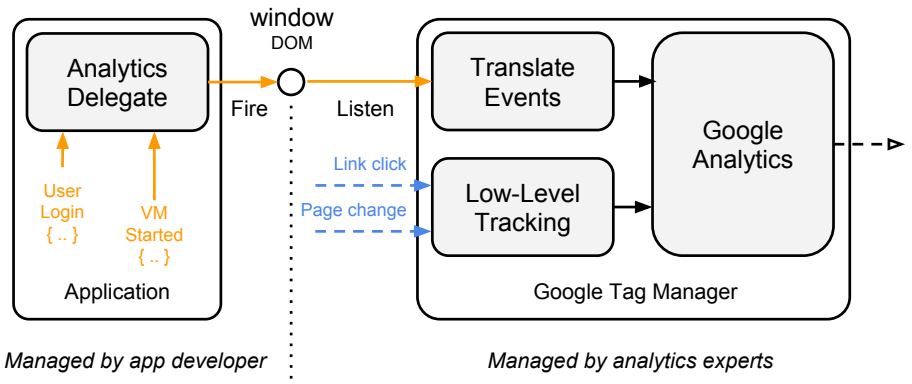


Figure 1. ***Event flow for an application using CCLTracker and Google Analytics. The* CCLTracker Javascript Library *is the* Analytics Delegate *(at the left)***

3.1.1. Persistent Store

In one of our CS projects, it was important to preserve the state across browsers or sessions. This was because the user was instructed to start a Virtual Machine and we wanted to count how many hours (s)he spent on it. Our solution was to check the state of the machine every time the user came back to the interface and if the VM was still running, accumulate the time since the last check. However, this introduced a problem, since we could not preserve the state information across different browsers.

To solve this problem, the *CCLTracker Javascript Library* is equipped with a state sharing mechanism. Thus, when the state of the library is changed, a callback is fired, allowing interested parties to push the change to a persistent store. Likewise, it provides an `importStore` function for importing a previously archived state.

In the case of the *CERN 60 Computing Challenge* (see below) we kept the persistent store in a Virtual Machine that was running in the user's computer. When the browser window was focused, the analytics store was retrieved from the VM. Likewise, when a change in the analytics store was triggered, the entire state was pushed back to the VM. This way, even if the user was changing browsers the entire analytics state was being kept.

A demonstration of this interface is shown below:

```
// ----------------------------------------------------------------
// Listen for permanent state changes from the analytics library.
// ----------------------------------------------------------------
$(window.analytics).on('changed', function(e, storeStr) {

    // Keep it somewhere you know is more permanent
    somePermanentStore.save( storeStr );

});
```

```
// ------------------------------------------------------
// In a similar manner, when the permanent data are known,
// you do the reverse operation
// ------------------------------------------------------
$(window).on('load', function(e) {
    // Restore the permanent data
    var data = somePermanentStore.restore();

    // Import to analytics
    analytics.importStore( data );
});
```

3.1.2. Assisting Utilities

In addition to the event tracking mechanism, the *CCLTracker Javascript Library* has a variety of built-in utilities for assisting the collection of metrics and other information.

First of all, **Stopwatches** are of particular use when you want to measure the time between two events. The *CCLTracker Javascript Library* provides named timers that you can start and stop on demand. For instance:

```
// ------------------------------------------------
// When the user focuses on the window, start the timer
// ------------------------------------------------
$(window).focus(function() {
    analytics.startTimer('idle-time');
});

// ------------------------------------------------------
// When the user moves focus away, fire an event specifying
// how much time the user spent on-line.-
// ------------------------------------------------------
$(window).blur(function() {
    var offTime = analytics.stopTimer('idle-time');
    analytics.fireEvent('user.online', {
        'time': offTime
    });
});
```

In addition the *CCLTracker Javascript Library* can calculate accumulated values or deltas, trying to use the previous values kept in the persistent store. This means that you can accumulate values across different visits. For example:

```
// Add 10 more hits
var hits = analytics.accumulate('ui.hits', 10);

// Count how many new jobs are processed
var new_jobs = analytics.delta('vm.jobs', get_job_count());
```

The *CCLTracker Javascript Library* will use the browser's `localStore` and the persistent mechanism explained in the previous section in order to correctly calculate the new values.

3.2. Using the library

From the developer's point-of-view, the moment the Javascript library is included, the global object `analytics` is available for use. That is the case even if there is no analytics back-end at the time, making the application completely agnostic to it.

The following snippet demonstrates a very simple analytics application that tracks views and clicks:

```
<script type="text/javascript" src="analytics.min.js"></script>
<script type="text/javascript">

    // -------------------------------------------------------------
    // The script is loaded, which means the user has seen the page
    // -------------------------------------------------------------
    analytics.fireEvent('pageview');

    // -----------------------------------------------------
    // Listen for window click events and send the coordinates
    // -----------------------------------------------------
    document.body.addEventListener('click', function(e) {
        analytics.fireEvent('click', {
            'x': e.offsetX,
            'y': e.offsetY
        })
    });

</script>
```

Once the application is broadcasting its events, an analytics expert can hook on the appropriate callbacks and receive its events. For instance, the developer can place a GTM tag and the analytics expert can then place his code at a later time:

```
    // ----------------------------------------
    // Listen for analytics events in the GTM tag
    // ----------------------------------------
    $(window).on('analytics.pageview', function(event, eventData) {
    // Forward event to GTM data layer
        dataLayer.push({'event': 'pageview'});
    });
    $(window).on('analytics.click', function(event, eventData) {
        // Forward event to GTM data layer
        dataLayer.push({
            'event': 'click',
            'x': eventData.x,
            'y': eventData.y
        });
    });

    // ----------------------------------------
    // It is important to let CCLTracker library
    // know that someone registered for receiving events.
    // ----------------------------------------
    window.analyticsListener = true;
```

The GTM code can be loaded at any time in the page. The *CCLTracker Javascript Library* will seamlessly use it the moment it becomes available, without losing any of the previously triggered events.

3.3. Example

In the case of the **CERN 60 Computing Challenge** (Section 5.2), we used a hybrid event tracking mechanism. We collected user behaviour information through Google Tag Manager, using high-level DOM events (such as link clicks), and fine-grained participation information through the *CCLTracker Javascript Library* .

In brief, when a user participates in the challenge, a Virtual Machine is installed in his computer and interfaced through the challenge web interface. This is achieved using the lightweight *CernVM WebAPI* [3] application, that acts as a bridge between the browser and the hypervisor. Upon a user's action, the simulation software is started inside the VM and he is presented in real-time with the results. At the same time, the user may choose to change the resources (s)he allocates to the project, or stop it completely. Our goal was to be able to track all these events using the *CCLTracker Javascript Library* .

The *CernVM WebAPI* javascript interface library exposes a variety of status events. For example it is possible to detect installation problems or other faults that affect the user experience. You can read more on Section 5.2, but we can briefly mention the following events:

- WebAPI Helper Installed/Started/Failure
- WebAPI Virtual Machine Created/Started/Paused/Resumed/Stopped/Failure
- Simulation Application (from within the VM) Status Updates/Failures

As you can see, using the *CernVM WebAPI* interface it is possible to track most of the usability problems. For instance, we can understand if a user managed to install *CernVM WebAPI*, created a VM and then started it, or if he failed to complete one of the steps. However the most interesting feature was the fact that we could receive updates from the running software and post it to the analytics platform in (near) real-time. Even if the user closed the browser window and then opened it again at a later time, the interface was able to calculate the differences and fire the correct analytics events.

To achieve this, it used the persistent store, keeping the analytics state in a property in the Virtual Machine. This way, even if the user was visiting the same interface from a different browser, it could still track his/her progress. The following code demonstrates this feature:

```
$(vmMonitor).on('simulationPropertyChanged',  function(e, property, value) {
    if (property == "jobs/completed") {
        // Count the new jobs since user's last visit
        var new_jobs = analytics.delta('vm.jobs', value);
        if (new_jobs > 0) {
            // Send that many 'jobs.completed' event
            for (var i=0; i<new_jobs; i++) {
                analytics.fireEvent('jobs.completed');
            }
        }
    }
});
```

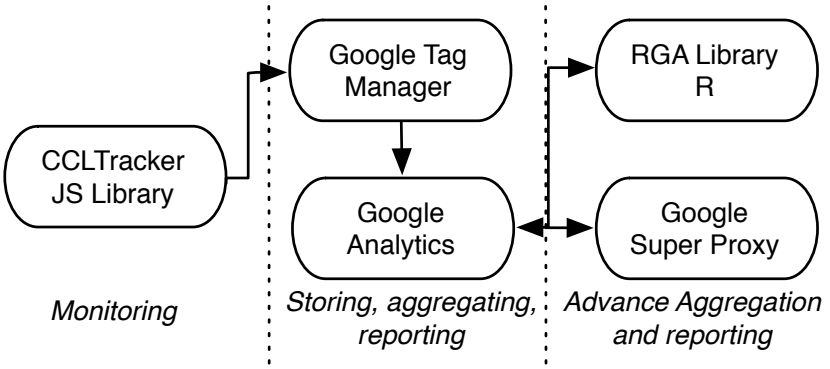


Figure 2. CCLTracker Framework Components

4. CCLTRACKER FRAMEWORK

The CCLTracker JavaScript Library has been integrated in a framework composed by the following tools and services:

Google Tag Manager (GTM) is in charge of structuring the analytics data and sending it to Google Analytics. GTM receives events generated by the CCLTracker Library that are fired from the website. GTM is used as a connector between the website and the analytics tool (Google Analytics in this case). By changing GTM settings we could address another analytics tool such as Piwik, or directly store the events in a local server without requiring any change in the website code.

Google Analytics provides storage, aggregation and visualization of data. Additionally Google Analytics provides web-based analytics information. Information about user activities and system performance is sent by Google Tag Manager.

R language is used to create advanced aggregation of analytics data. It allows us to measure engagement in many different ways, and to cluster user profiles.

Google Super Proxy allows us to publish analytics data publicly.

Google Query Explorer is an online service used to retrieve analytics data from Google Analytics. The queries created in Google Query Explorer are used in R for advance aggregation and Google Super Proxy for making data publicly available.

One of the major decisions taken in this framework was the use of Google Analytics. Google Analytics was adopted as analytics tool because of the demographics information provided, the fast set-up time, the free service, and the large user community surrounding it.

Figure 2 shows the interaction between the different components of the CCLTracker Framework.

5. PROOF OF CONCEPT

This section summarizes the main applications where the CCLTracker framework has been used and shows the potential of using CCLTracker for measuring engagement and learning in citizen science projects.

[3]https://github.com/wavesoft/cernvm-webapi/

The CCLTracker framework has been implemented under the EU project Citizen CyberLab. The main goal was to evaluate the different pilots developed during the project regarding user engagement, outreach actions, and user learning outcomes.

CCLTracker has been used by the following pilots:

- **Virtual Atom Smasher**[4] is an educational interactive game that teaches users about particle physics, while at the same time aiding theoretical physicists at CERN with their research.
- **CERN Volunteer Computing Challenge**[5] is a web site running periodic public volunteer computing challenges. Volunteers help CERN scientists simulate particle collisions in accelerators like the Large Hadron Collider (LHC), using their own computers.

The CCLTracker Framework has been integrated in these applications in order to understand the user behaviour, measuring engagement and learning.

5.1. Virtual Atom Smasher Game

Virtual Atom Smasher is an educational game that allows citizens to contribute to scientific discoveries at CERN without any prior knowledge of particle physics. Participants control a Monte Carlo event generator (a virtual particle collider) that produces scientific data analogous to that generated by real High Energy Physics experiments, and their goal is to tune its parameters until the results match the measured experimental results. The outcomes are important to the scientific community, since they can validate current scientific hypotheses.

Monitoring user actions with CCLTracker allows us to measure the percentage of users who validate new parameter values. We consider “visitors" those users who come to the game but do not validate any parameter values, and “participants" those users who change parameter values and validate them. Analytics data shows that 40% of users validated new parameter values, i.e. our public was made up of 60% visitors and 40% participants. We consider visitors as users with the lowest level of engagement, while participants shows higher engagement levels.

Additionally, different higher level of engagement were measured by monitoring the number of validations carried out by the participants (see Figure 3). Analogously to other citizen science projects we observed that a small number of users carried out the majority of validations.

To see whether the users learnt the game dynamics and concepts, we focused on two segments of users: those who followed the expected flow of actions and those who did not (i.e. they were not playing as expected). Namely, the expected flow of actions is: (1) change parameter values, (2) click estimate button, (3) repeat 1 and 2, and finally (4) validate.

Figure 4 shows the number of users per day who were following the expected flow of actions (orange line) versus the number of users per day who did not follow the expected flow (blue line). We observe that on the 17th of June and 15th of July users played the game properly. On those days, the VAS lead developer was giving seminars at CERN introducing the game and helping new users to play it. Consequently, users were playing properly for a few days after the seminar. Later, new users arrived and they had problems to find out the right way of playing. Based on this information revealed by the CCLTracker framework, the VAS user interface was re-designed in order to ease participation.

[4] http://test4theory.cern.ch/about/

[5] https://test4theory.cern.ch/challenge/

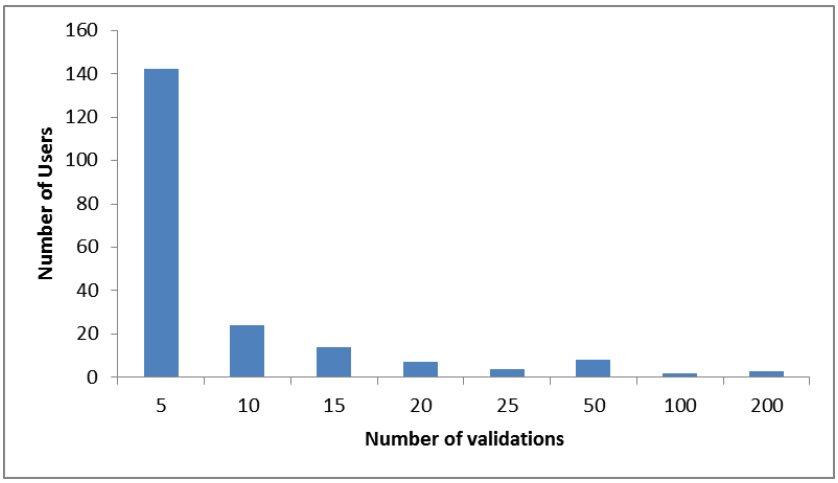


Figure 3. *VAS: Number of users per number of validations*

Figure 4. *VAS: participants who are playing according to the expected flow of actions (orange line) versus those who are not (blue line)*

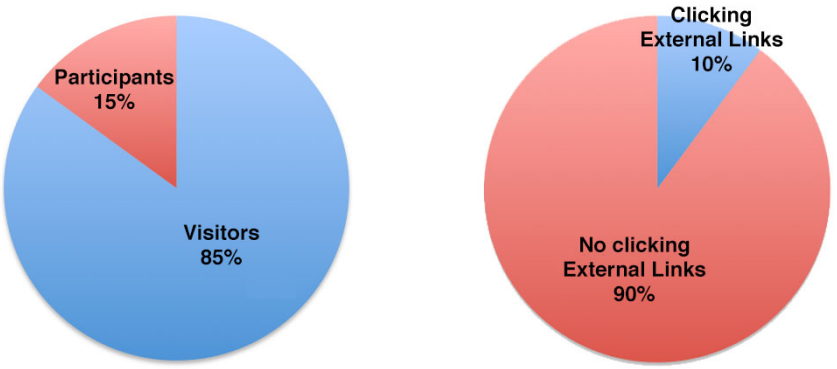
5.2. CERN Volunteer Computing Challenges

The CERN Volunteer Computing Challenge is a periodic volunteer computing event. The CCLTracker framework has been used in two of those events, namely the CERN 60 Challenge, a 12-day event that ran in December 2014 (9-20 Dec.), and the CERN Public Computing Challenge 2015 that ran from November 1 to December 1 2015. The purpose of these challenges was to test the experimental CernVM WebAPI technology and to understand user engagement in the volunteer computing project by using analytics data. The CERN VM technology allows volunteers all around the world to easily contribute in volunteer computing projects by donating computational resources from their personal computers. It is based on Virtual Machine technology which allows volunteers to easily contribute independently of the operating system they use.

The events monitored by the CCLTracker framework can be classified into 3 general groups:

User actions contains all events related to user actions on the challenge website such as clicking external links, start button, stop button, or changing the virtual machine settings (i.e. the number of cores and memory the user is donating).

User goals: are used to monitor user achievements during the participation on the challenge. Some examples are CPU time donated, number of jobs computed, or how many times a user starts the VM. Goals are pre-aggregated before firing the event. This has two main advantages: (1) it eases the data analysis and (2) demographic data can be extracted for different user segments based on participation (e.g. gender of users



(a) (b)

Figure 5. ***CERN Challenge participation (a) % of participants and visitors, and (b) % of users clicking external links***

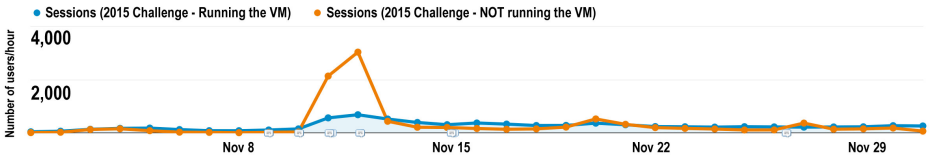


Figure 6. ***CERN Challenge: Sessions of users not running the VM versus sessions running the VM***

who contribute more than 100 jobs, or interest of users who visit the website but do not contribute).

VM status: contains all events related to the status of the virtual machine, i.e. available, paused, running, stopped, or booted.

WebAPIStatus: contains events related to the web interface to the Virtual Machine.

Google Analytics shows the total number of sessions and users however it is not able to provide information about the participation of the users. Using CCLTracker we can distinguish between visitors (i.e. users who visit the site but they do not contribute to the challenge) and participants (i.e. those users who install the virtual machine and share computational resources). As example Figure 5 shows the percentage of visitors and participants and the percentage of users clicking external links.

Figure 6 shows the number of sessions per day coming from visitors and participants. The blue line represents visitors and the orange represents participants. We could identify the peaks generated by different outreach actions, and the percentage of participants engaged by these actions. In this way we can measure the impact of outreach actions beyond simply the number of visitors, e.g. how many participants we gathered for each outreach action.

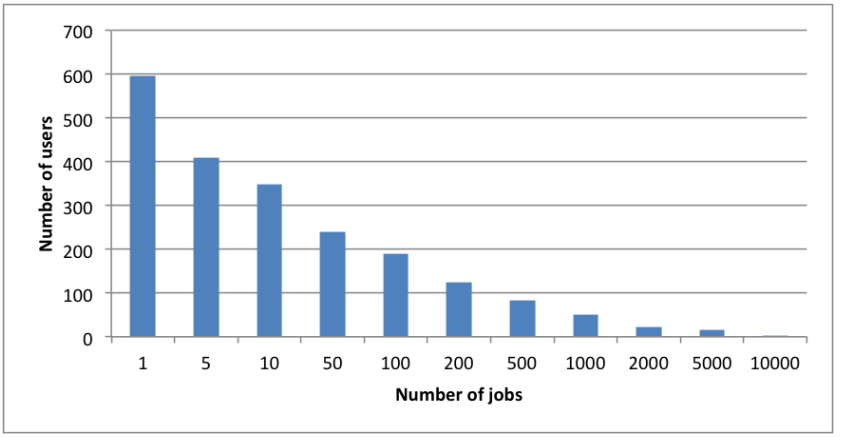


Figure 7. CERN Challenge: Number of users by level of participation (number of jobs processed)

We established different levels of participation in order to measure user engagement and get web-based analytics information such as referrals, demographic information, or interest for the different participation levels. Figure 7 shows the number of users for the different participation levels, i.e. at least 1 job, 5 jobs, 10 jobs, ...

The number of jobs computed could be subject to the computational resources of the user machine. Thus, the level of participation could be computed also by the CPU time donated to the event. Figure 8 shows the number of users for each participation level based on number of hours of CPU processing.

CCLTracker also allows crosschecking user location with participation information, thus we can see the participation from each county and the participation level (i.e. how much each user contributed) of the users coming from each country (see Table 1).

Additionally, we can evaluate the participation of users based on the different referrals (see Table 2). This is especially relevant for measuring outreach actions and improving user engagement.

CCLTracker allows combining demographic information from Google Analytics with monitoring information coming from the CCLTracker JavaScript library. Google Analytics provides gender and age information about the users who have Google account, i.e. the information is gathered from the Google account profile. Google analytics is able to provide users' gender and age for more than 50% of the visitors. CCLTracker also allows to see the evolution of gender balance for different levels of participation. Figure 9 shows the gender balance for users who do not compute any job, at least one job, more than 10 jobs and finally more than 100 jobs. This information is very relevant to find out who are the major contributors in your project, thus improving the outreach activities.

6. CONCLUSION

This paper presents the CCLTracker framework for measuring user engagement and learning in citizen science projects. There are two main contributions: (1) A JavaScript library for monitoring user activities in citizen science projects, and (2) the integration into an analytics framework for combining the user activities with web-based analytics information and tools. We present some analytics results as a proof of concept to highlight that web-based analytics information such as the information provided by Google Analytics does

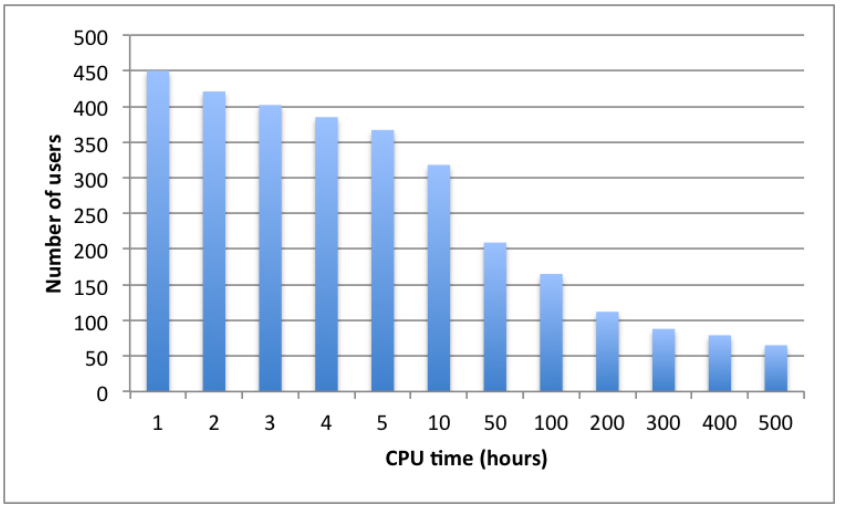
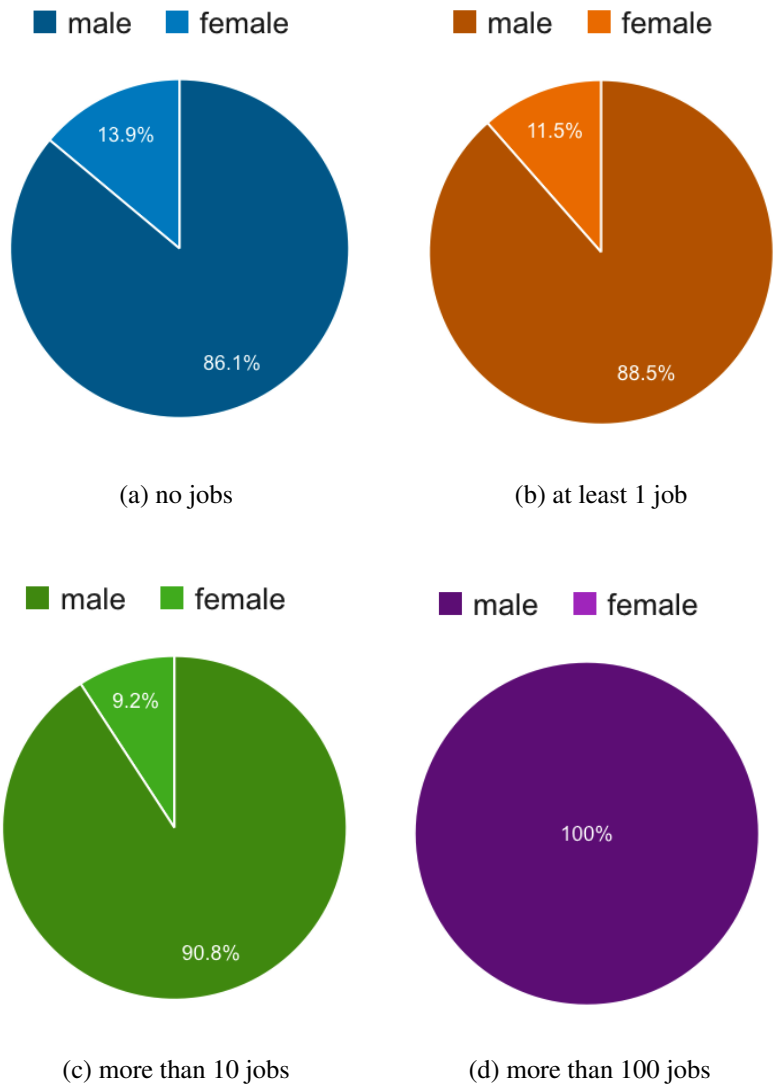


Figure 8. ***CERN Challenge: Number of users by level of participation (CPU time donated)***

Country	Number of jobs	Sessions	% Sessions
USA	no jobs	2411	22.23%
	+1 job	1346	20.44%
	+10 jobs	1071	22.03%
	+100 jobs	892	29.97%
UK	no jobs	795	7.33%
	+1 job	390	5.92%
	+10 jobs	223	4.59%
	+100 jobs	129	4.33%
India	no jobs	615	5.67%
	+1 job	29	0.44%
	+10 jobs	21	0.43%
	+100 jobs	21	0.71%
Germany	no jobs	607	5.60%
	+1 job	862	13.09%
	+10 jobs	669	13.76%
	+100 jobs	452	15.19%

Table 1. ***CERN Challenge: Participation by country***



(a) no jobs

(b) at least 1 job

(c) more than 10 jobs

(d) more than 100 jobs

Figure 9. ***CERN Challenge: Gender balance among users. (a) users not computing any job, (b) users computing at least 1 jobs, (c) users computing more than 10 jobs, and (d) users computing more than 100 jobs***

Referral	Number of jobs	Sessions	% Sessions
reddit.com	no jobs	1492	30.27%
	+1 job	206	9.43%
	+10 jobs	127	7.84%
	+100 jobs	57	5.78%
news.ycombinator.com	no jobs	1419	28.79%
	+1 job	62	2.84%
	+10 jobs	39	2.41%
	+100 jobs	21	2.13%
Facebook	no jobs	684	13.88%
	+1 job	402	18.40%
	+10 jobs	271	16.74%
	+100 jobs	148	15.01%

Table 2. CERN Challenge: Number of participants by referral

not satisfy the requirements for measuring user engagement and learning, since the information it provides about user behaviour is very poor (i.e. average time per session, or number of page views) and to show the potential of the CCLTracker JavaScript library when integrated into the CCLTracker analytics framework.

While this paper focuses on the technical aspect of the CCLTracker, a deeper analysis of potential key learning indicators using CCLTracker in Citizen Science project is presented in Oula Abu-Amsha et al. (Abu-Amsha, Schneider, Fernandez-Marquez, Costa, Fuchs, and Kloetzer, Abu-Amsha et al.).

7. FUTURE WORK

One of the main challenges when implementing user monitoring features in a web-based application is to keep the monitoring code stable over different versions of the application. Often, implementation of new functionality produces either untracked user actions or wrong monitoring of data. Even when all programmers are aware of the monitoring system, there is not a systematic way of validating the right behaviour of the tracking code. For debugging the CCLTracker implementation on the two projects presented in this paper, we have used the realtime feature of Google Analytics. However, this requires checking when new functionalities or major changes are carried out on the application code. An interesting improvement would be to implement unit testing with tools such as PhantomJS[6] or QUnit [7] to automatically verify that the monitoring code is working properly afterwards.

8. ACKNOWLEDGMENTS

This work has been supported by the EU-FP7 STREP project Citizen Cyberlab: Technology Enhanced Creative Learning in the field of Citizen Cyberscience, under the contract no. 317705.

[6]http://phantomjs.org/

[7]https://qunitjs.com/

9. REFERENCES

The 1st International Conference on Learning Analytics and Knowledge. http://tekri.athabascau.ca/analytics/. (????). [Online; accessed 10-Dec-2017].

"Google Analytics". http://www.google.com/analytics/standard/. (????). [Online; accessed 20-Dec-2016].

Piwik:Open Analytics Platform. http://piwik.org/. (????). [Online; accessed 20-Dec-2016].

Usage of traffic analysis tools for websites. https://w3techs.com/technologies/overview/traffic_analysis/all. (????). [Online; accessed 10-Jan-2017].

Abu-Amsha, O, Schneider, D. K, Fernandez-Marquez, J. L, Costa, J. D, Fuchs, B, and Kloetzer, L. Learning Analytics in Citizen Cyberscience: A Framework to Evaluate Participant Learning and Engagement with Analytics. *Human Computation* 3, 1 (????).

Ferguson, R. (2012). Learning analytics: drivers, developments and challenges. *International Journal of Technology Enhanced Learning* 4, 5-6 (2012), 304–317.

Kloetzer, L, Schneider, D. K, and Costa, J. D. (2017). Not So Passive: Engagement and Learning in Volunteer Computing. *Human Computation* 3, 1 (2017).

Miller, S. A. (2012). *Piwik Web Analytics Essentials*. Packt Publishing Ltd.

O'Brien, H and Toms, E. (2008). What is user engagement? A conceptual framework for defining user engagement with technology. *Journal of the American Society for Information Science and Technology* 59, 6 (2008), 938–955.

Ponciano, L and Brazileiro, F. (2014). Finding Volunteers' Engagement Profiles in Human Computation for Citizen Science Projects. *Human Computation* 1, 3 (2014).