SensSoftProposal

SensSoft Proposal

Abstract

The Software as a Sensor™ (SensSoft) Project offers an open-source (ALv2.0) software tool usability testing platform. It includes a number of components that work together to provide a platform for collecting data about user interactions with software tools, as well as archiving, analyzing and visualizing that data. Additional components allow for conducting web-based experiments in order to capture this data within a larger experimental framework for formal user testing. These components currently support Java Script-based web applications, although the schema for “logging” user interactions can support mobile and desktop applications, as well. Collectively, the Software as a Sensor Project provides an open source platform for assessing how users interacted with technology, not just collecting what they interacted with.

Proposal

The Software as a Sensor™ Project is a next-generation platform for analyzing how individuals and groups of people make use of software tools to perform tasks or interact with other systems. It is composed of a number of integrated components:

- User Analytic Logging Engine (User ALE) refers to a simple Application Program Interface (API) and backend infrastructure. User ALE provides “instrumentation” for software tools, such that each user interaction within the application can be logged, and sent as a JSON message to an Elasticsearch/Logstash/Kibana (Elastic Stack) backend.
    - The API provides a robust schema that makes user activities human readable, and provides an interpretive context for understanding that activity’s functional relevance within the application. The schema provides highly granular information best suited for advanced analytics. This hierarchical schema is as follows:
        - Element Group: App features that share function (e.g., map group)
        - Element Sub: Specific App feature (e.g., map tiles)
        - Element Type: Category of feature (e.g., map)
        - Element ID: [attribute] id
        - Activity: Human imposed label (e.g., “search”)
        - Action: Event class (e.g., zoom, hover, click)
    - The API can either be manually embedded in the app source code, or implemented automatically by inserting a script tag in the source code.
    - Users can either setup up their own Elastic stack instance, or use Vagrant, a virtualization environment, to deploy a fully configured Elastic stack instance to ship and ingest user activity logs and visualize their log data with Kibana.
    - RESTful APIs allow other services to access logs directly from Elasticsearch.
    - User ALE allows adopters to own the data they collect from users outright, and utilize it as they see fit.
- Distill is an analytics stack for processing user activity logs collected through User ALE. Distill is fully implemented in Python, dependent on graph-tool to support graph analytics and other external python libraries to query Elasticsearch. The two principle functions of Distill are segmentation and graph analytics:
    - Segmentation allows for partitioning of the available data along multiple axes. Subsets of log data can be selected via their attributes in User ALE (e.g. Element Group or Activity), and by users/sessions. Distill also has the capability to ingest and segment data by additional attributes collected through other channels (e.g. survey data, demographics).This allows adopters to focus their analysis of log data on precisely the attributes of their app (or users) they care most about.
    - Distill’s usage metrics are derived from a probabilistic representation of the time series of users’ interactions with the elements of the application. A directed network is constructed from the representation, and metrics from graph theory (e.g. betweenness centrality, in/out-degree of nodes) are derived from the structure. These metrics provide adopters ways of understanding how different facets of the app are used together, and they capture canonical usage patterns of their application. This broad analytic framework provides adopters a way to develop and utilize their own metrics
- The Test Application Portal (TAP) provides a single, user-friendly interface to Software as a Sensor™ Project components, including visualization functionality for Distill Outputs leveraging Django, React, and D3.js. It has two key functions:
    - It allows adopters to register apps, providing metadata regarding location, app name, version, etc., as well as permissions regarding who can access user data. This information is propagated to all other components of the larger system.
    - The portal also stages visualization libraries that make calls to Distill. This allows adopters to analyze their data as they wish to; it’s “dashboard” feel provides a way to customize their views with adopter-generated widgets (e.g., D3 libraries) beyond what is included in the initial open source offering.
- The Subject Tracking and Online User Testing (STOUT) application is an optional component that turns Software as a Sensor™ Technology into a research/experimentation enterprise. Designed for psychologists and HCI/UX researchers, STOUT allows comprehensive human subjects data protection, tracking, and tasking for formal research on software tools. STOUT is primarily python, with Django back-end for authentication, permissions, and tracking, MongoDB for databasing, and D3 for visualization. STOUT includes a number of key features:
    - Participants can register in studies of software tools using their own preferred credentials. As part of registration, participants can be directed through human subjects review board compliant consent forms before study enrollment.
    - STOUT stores URLs to web/network accessible software tools as well as URLs to third party survey services (e.g., surveymonkey), this allows adopters to pair software tools with tasks, and collect survey data and comments from participants prior to, during, or following testing with software tools.
    - STOUT tracks participants’ progress internally, and by appending a unique identifier, and task identifier to URLs. This information can be passed to other processes (e.g., User ALE) allowing for disambiguation between participants and tasks in experiments on the open web.
    - STOUT supports between and within-subjects experimental designs, with random assignment to experimental conditions. This allows for testing across different versions of applications.
    - STOUT can also use Django output (e.g., task complete) to automate other processes, such as automated polling applications serving 3rd party form data APIs (e.g.,SurveyMonkey), and python or R scripts to provide automated post-processing on task or survey data.
    - STOUT provides adopters a comprehensive dashboard view of data collected and post-processed through its extensions; in addition to user enrollment, task completion, and experiment progress metrics, STOUT allows adopters to visualize distributions of scores collected from task and survey data.

Each component is available through its own repository to support organic growth for each component, as well as growth of the whole platform's capabilities.

Background and Rationale

Any tool that people use to accomplish a task can be instrumented; once instrumented, those tools can be used to report how they were used to perform that task. Software tools are ubiquitous interfaces for people to interact with data and other technology that can be instrumented for such a purpose. Tools are different than web pages or simple displays, however; they are not simply archives for information. Rather, they are ways of interfacing with and manipulating data and other technology. There are numerous consumer solutions for understanding how people move through web pages and displays (e.g., Google Analytics, Adobe Omniture). There are far fewer options for understanding how software tools are used. This requires understanding how users integrate a tool's functionality into usage strategies to perform tasks, how users sequence the functionality provided them, and deeper knowledge of how users understand the features of software as a cohesive tool. The Software as a Sensor™ Project is designed to address this gap, providing the public an agile, cost-efficient solution for improving software tool design, implementation, and usability.

Software as a Sensor™ Project Overview

? Unknown Attachment

Figure 1. User ALE Elastic Back End Schema, with Transfer Protocols.

Funded through the DARPA XDATA program and other sources, the Software as a Sensor™ Project provides an open source (ALv2.0) solution for instrumenting software tools developed for the web so that when users interact with it, their behavior is captured. User behavior, or user activities, are captured and time-stamped through a simple application program interface (API) called User Analytic Logging Engine (User ALE). User ALE's key differentiator is the schema that it uses to collect information about user activities; it provides sufficient context to understand activities within the software tool's overall functionality. User ALE captures each user initiated action, or event (e.g., hover, click, etc.), as a nested action within a specific element (e.g., map object, drop down item, etc.), which are in turn nested within element groups (e.g., map, drop down list) (see Figure 1). This information schema provides sufficient context to understand and disambiguate user events from one another. In turn, this enables myriad analysis possibilities at different levels of tool design and more utility to end-user than commercial services currently offer. Once instrumented with User ALE, software tools become human signal sensors in their own right. Most importantly, the data that User ALE collects is owned outright by adopters and can be made available to other processes through scalable Elastic infrastructure and easy-to-manage Restful APIs. Distill is the analytic framework of the Software as a Sensor™ Project, providing (at release) segmentation and graph analysis metrics describing users' interactions with the application to adopters. The segmentation features allow adopters to focus their analyses of user activity data based on desired data attributes (e.g., certain interactions, elements, etc.), as well as attributes describing the software tool users, if that data was also collected. Distill's usage and usability metrics are derived from a representation of users' sequential interactions with the application as a directed graph. This provides an extensible framework for providing insight as to how users integrate the functional components of the application to accomplish tasks.

? Unknown Attachment

Figure 2. Software as a Sensor™ System Architecture with all components.

The Test Application Portal (TAP) provides a single point of interface for adopters of the Software as a Sensor™ project. Through the Portal, adopters can register their applications, providing version data and permissions to others for accessing data. The Portal ensures that all components of the Software as a Sensor™ Project have the same information. The Portal also hosts a number of python D3 visualization libraries, providing adopters with a customizable "dashboard" with which to analyze and view user activity data, calling analytic processes from Distill. Finally, the Subject Tracking and Online User Testing (STOUT) application, provides support for HCI/UX researchers that want to collect data from users in systematic ways or within experimental designs. STOUT supports user registration, anonymization, user tracking, tasking (see Figure 3), and data integration from a variety of services. STOUT allows adopters to perform human subject review board compliant research studies, and both between- and within-subjects designs. Adopters can add tasks, surveys and questionnaires through 3rd party services (e.g., SurveyMonkey). STOUT tracks users' progress by passing a unique user IDs to other services, allowing researchers to trace progress by passing a unique user IDs to other services, allowing researchers to trace form data and User ALE logs to specific users and task sets (see Figure 4).

? Unknown Attachment

Figure 3. STOUT assigns participants subjects to experimental conditions and ensures the correct task sequence. STOUT's Django back end provides data on task completion, this can be used to drive other automation, including unlocking different task sequences and/or achievements.

? Unknown Attachment

Figure 4. STOUT User Tracking. Anonymized User IDs (hashes) are concatenated with unique Task IDs. This "Session ID" is appended to URLs (see Highlighted region), custom variable fields, and User ALE, to provide and integrated user testing data collection service.

STOUT also provides for data polling from third party services (e.g., SurveyMonkey) and integration with python or R scripts for statistical processing of data collected through STOUT. D3 visualization libraries embedded in STOUT allow adopters to view distributions of quantitative data collected from form data (see Figure 5).

? Unknown Attachment

Figure 5. STOUT Visualization. STOUT gives experimenters direct and continuous access to automatically processed research data.

Insights from User Activity Logs

The Software as a Sensor™ Project provides data collection and analytic services for user activities collected during interaction with software tools. However, the Software as a Sensor™ Project emerged from years of research focused on the development of novel, reliable methods for measuring individuals' cognitive state in a variety of contexts. Traditional approaches to assessment in a laboratory setting include surveys, questionnaires, and physiology (Poore et al., 2016). Research performed as part of the Software as a Sensor™ project has shown that the same kind of insights derived from these standard measurement approaches can also be derived from users' behavior. Additionally, we have explored insights that can only be gained by analyzing raw behavior collected through software interactions (Mariano et al., 2015). The signal processing and algorithmic approaches resulting from this research have been integrated into the Distill analytics stack. This means that adopters will not be left to discern for themselves how to draw insights from the data they gather about their software tools, although they will have the freedom to explore their own methods as well. Insights from user activities provided by Distill's analytics framework fall under two categories, broadly classified as functional workflow and usage statistics:
Functional workflow insights tell adopters how user activities are connected, providing them with representations of how users integrate the application's features together in time. These insights are informative for understanding the step-by-step process by which users interact with certain facets of a tool. For example, questions like "how are my users, constructing plots?" are addressable through workflow analysis. Workflows provide granular understanding of process level mechanics and can be modeled probabilistically through a directed graph representation of the data, and by identification of meaningful sub-sequences of user activities actually observed in the population. Metrics derived provide insight about the structure and temporal features of these mechanics, and can help highlight efficiency problems within workflows. For example, workflow analysis could help identify recursive, repetitive behaviors, and might be used to define what "floundering" looks like for that particular tool. Functional workflow analysis can also support analyses with more breadth. Questions like, "how are my users integrating my tools' features into a cohesive whole? Are they relying on the tool as a whole or just using very specific parts of it?" Adopters will be able to explore how users think about software as cohesive tools and examine if users are relying on certain features as central navigation or analytic features. This allows for insights into whether tools are designed well enough for users to understand that they need to rely on multiple features together. Through segmentation, adopters can select the subset of the data ~~software element, action, user demographics, geographic location, etc.~~ they want to analyze. This will allow them to compare, for example, specific user populations against one another in terms of how they integrate software functionality. Importantly, the graph-based analytics approach provides a flexible representation of the time series data that can capture and quantify canonical usage patterns, enabling direct comparisons between users based on attributes of interest. Other modeling approaches have been utilized to explore similar insights and may be integrated at a later date (Mariano, et al., 2015). Usage statistics derive metrics from simple frequentist approaches to understanding, coarsely, how much users are actually using applications. This is different from simple "traffic" metrics, however, which assess how many users are navigating to a page or tool. Rather usage data provides insight on how much raw effort (e.g., number of activities) is being expended while users are interacting with the application. This provides deeper insight into discriminating "visitors" from "users" of software tools. Moreover, given the information schema User ALE provides, adopters will be able to delve into usage metrics related to specific facets of their application. Given these insights, different sets of adopters—software developers, HCI/UX researchers, and project managers—may utilize The Software as a Sensor™ Project for a variety different use cases, which may include:

- Testing to see if users are interacting with software tools in expected or unexpected ways.
- Understanding how much users are using different facets of different features in service of planning future developments.
- Gaining additional context for translating user/customer comments into actionable software fixes.
- Understanding which features users have trouble integrating to guide decisions on how to allocate resources to further documentation.
- Understanding the impact that new developments have on usability from version to version.
- Market research on how users make use of competitors' applications to guide decisions on how to build discriminating software tools.
- General research on Human Computer Interaction in service of refining UX and design principles.
- Psychological science research using software as data collection platforms for cognitive tasks.

Differentiators

The Software as a Sensor™ Project is ultimately designed to address the wide gaps between current best practices in software user testing and trends toward agile software development practices. Like much of the applied psychological sciences, user testing methods generally borrow heavily from basic research methods. These methods are designed to make data collection systematic and remove extraneous influences on test conditions. However, this usually means removing what we test from dynamic, noisy—real-life—environments. The Software as a Sensor™ Project is designed to allow for the same kind of systematic data collection that we expect in the laboratory, but in real-life software environments, by making software environments data collection platforms. In doing so, we aim to not only collect data from more realistic environments, and use-cases, but also to integrate the test enterprise into agile software development process. Our vision for The Software as a Sensor™ Project is that it provides software developers, HCI/UX researchers, and project managers a mechanism for continuous, iterative usability testing for software tools in a way that supports the flow (and schedule) of modern software development practices—Iterative, Waterfall, Spiral, and Agile. This is enabled by a few discriminating facets:

? Unknown Attachment

Figure 6. Version to Version Testing for Agile, Iterative Software Development Methods. The Software as a Sensor™ Project enables new methods for collecting large amounts of data on software tools, deriving insights rapidly to inject into subsequent iterations

- Insights enabling software tool usability assessment and improvement can be inferred directly from interactions with the tool in "real-world" environments. This is a sea-change in thinking compared to canonical laboratory approaches that seek to artificially isolate extraneous influences on the user and the software. The Software as a Sensor™ Project enables large scale, remote, opportunities for data collection with minimal investment and no expensive lab equipment (or laboratory training). This allows adopters to see how users will interact with their technology in their places of work, at home, etc.
- Insights are traceable to the software itself. Traditionally laboratory measures—questionnaires, interviews, and physiology—collect data that is convenient for making inferences about psychological states. However, it is notoriously difficult to translate this data into actionable "get-well" strategies in technology development. User ALE's information schema is specifically designed to dissect user interaction within the terminology of application design, providing a familiar nomenclature for software developers to interpret findings with.
- Granular data collection enables advanced modeling and analytics. User ALE's information schema dissects user interaction by giving context to activity within the functional architecture of software tools. Treating each time-series of user activity as a set of events nested within functional components provides sufficient information for a variety of modeling approaches that can be used to understand user states (e.g., engagement and cognitive load), user workflows (e.g., sub-sequences), and users' mental models of how software tool features can be integrated (in time) to

perform tasks. In contrast, commercial services such as Google Analytics and Adobe Analytics (Omniture) provide very sparse options for describing events. They generally advocate for using “boiler plate” event sets that are more suited to capturing count data for interactions with specific content (e.g., videos, music, banners) and workflows through “marketplace” like pages. User ALE provides content agnostic approaches for capturing user activities by letting adopters label them in domain specific ways that give them context. This provides a means by which identical user activities (e.g. click, select, etc.) can be disambiguated from each other based on which functional sub-component of the tool they have been assigned to.

- Adopter-generated content, analytics and data ownership. The Software as a Sensor™ Project is a set of open-source products built from other open-source products. This project will allow adopters to generate their own content easily, using open source analytics and visualization capabilities. By design, we also allow adopters to collect and manage their own data with support from widely used open source data architectures (e.g., Elastic). This means that adopters will not have to pay for additional content that they can develop themselves to make use of the service, and do not have to expose their data to third party commercial services. This is useful for highly proprietary software tools that are designed to make use of sensitive data, or are themselves sensitive.

Current Status

All components of the Software as a Sensor™ Project were originally designed and developed by Draper as part of DARPA’s XDATA project, although User ALE is being used on other funded R&D projects, including DARPA RSPACE, AFRL project, and Draper internally funded projects. Currently, only User ALE is publically available, however, the Portal, Distill, and STOUT will be publically available in the May/June 2016 time-frame. The last major release of User ALE was May, 2015. All components are currently maintained in separate repositories through GitHub (github.com/draperlaboratory). Currently, only software tools developed with Javascript are supported. However, we are currently working on pythonQT implementations for User ALE that will support many desktop applications.

Meritocracy

The current developers are familiar with meritocratic open source development at Apache. Apache was chosen specifically because we want to encourage this style of development for the project.

Community

The Software as a Sensor™ Project is new and our community is not yet established. However, community building and publicity is a major thrust. Our technology is generating interest within industry, particularly in the HCI/UX community, both Aptima and Charles River Analytics, for example are interested in being adopters. We have also begun publicizing the project to software development companies and universities, recently hosting a public focus group for Boston, MA area companies. We are also developing communities of interested within the DoD and Intelligence community. The NGA Xperience Lab has expressed interest in becoming a transition partner as has the Navy’s HCIL group. We are also aggressively pursuing adopters at AFRL’s Human Performance Wing, Analyst Test Bed. During incubation, we will explicitly seek to increase our adoption, including academic research, industry, and other end users interested in usability research.

Core Developers

The current set of core developers is relatively small, but includes Draper full-time staff. Community management will very likely be distributed across a few full-time staff that have been with the project for at least 2 years. Core personnel can be found on our website: http://www.draper.com/softwareasasensor

Alignment

Known Risks

Orphaned products

There are currently no orphaned products. Each component of The Software as a Sensor™ Project has roughly 1-2 dedicated staff, and there is substantial collaboration between projects.

Inexperience with Open Source

Draper has a number of open source software projects available through www.github.com/draperlaboratory.

Relationships with Other Apache Products

Software as a Sensor™ Project does not currently have any dependences on Apache Products. We are also interested in coordinating with other projects including Usergrid, and others involving data processing at large scales, time-series analysis and ETL processes.

Developers

The Software as a Sensor™ Project is primarily funded through contract work. There are currently no “dedicated” developers, however, the same core team does work will continue work on the project across different contracts that support different features. We do intend to maintain a core set of key personnel engaged in community development and maintenance—in the future this may mean dedicated developers funded internally to support the project, however, the project is tied to business development strategy to maintain funding into various facets of the project.

Documentation

Documentation is available through Github; each repository under the Software as a Sensor™ Project has documentation available through wiki’s attached to the repositories.

Initial Source

Current source resides at Github:

- https://github.com/draperlaboratory/user-ale (User ALE)
- https://github.com/draperlaboratory/distill (Distill)
- https://github.com/draperlaboratory/stout (STOUT and Extensions)
- https://github.com/draperlaboratory/

External Dependencies

Each component of the Software as a Sensor™ Project has its own dependencies. Documentation will be available for integrating them.

User ALE

- Elasticsearch: https://www.elastic.co/
- Logstash: https://www.elastic.co/products/logstash
- Kibana (optional): https://www.elastic.co/products/kibana

STOUT

- Django: https://www.djangoproject.com/
    - django-axes
    - django-custom-user
    - django-extensions
- Elasticsearch: https://www.elastic.co/
- Gunicorn: http://gunicorn.org/
- MySQL-python: https://pypi.python.org/pypi/MySQL-python
- Numpy: http://www.numpy.org/
- Pandas: http://pandas.pydata.org/
- psycopg2: http://initd.org/psycopg/
- pycrypto: https://www.dlitz.net/software/pycrypto/
- pymongo: https://api.mongodb.org/python/current/
- python-dateutil: https://labix.org/python-dateutil
- pytz: https://pypi.python.org/pypi/pytz/
- requests: http://docs.python-requests.org/en/master/
- six: https://pypi.python.org/pypi/six
- urllib3: https://pypi.python.org/pypi/urllib3
- mongoDB: https://www.mongodb.org/
- R (optional): https://www.r-project.org/

Distill

- Flask: http://flask.pocoo.org/
- Elasticsearch-dsl: https://github.com/elastic/elasticsearch-dsl-py
- graph-tool: https://git.skewed.de/count0/graph-tool
- OpenMp: http://openmp.org/wp/
- pandas: http://pandas.pydata.org/
- numpy: http://www.numpy.org/
- scipy: http://www.numpy.org/

Portal

- Django: https://www.djangoproject.com/
- React: https://facebook.github.io/react/
- D3.js: https://d3js.org/

GNU GPL 2

LGPL 2.1

Apache 2.0

GNU GPL

Required Resources

- Mailing Lists
  - private@senssoft.incubator.apache.org
  - dev@senssoft.incubator.apache.org
  - commits@senssoft.incubator.apache.org
- Git Repos
  - https://git-wip-us.apache.org/repos/asf/User-ALE.git
  - https://git-wip-us.apache.org/repos/asf/STOUT.git
  - https://git-wip-us.apache.org/repos/asf/DISTILL.git
  - https://git-wip-us.apache.org/repos/asf/TAP.git
- Issue Tracking
  - JIRA SensSoft (SENSSOFT)
- Continuous Integration
  - Jenkins builds on https://builds.apache.org/
- Web
  - http://SoftwareasaSensor.incubator.apache.org/
  - wiki at http://cwiki.apache.org

Initial Committers

The following is a list of the planned initial Apache committers (the active subset of the committers for the current repository on Github).

- Joshua Poore (jpoore@draper.com)
- Laura Mariano (lmariano@draper.com)
- Clayton Gimenez (cgimenez@draper.com)
- Alex Ford (aford@draper.com)
- Steve York (syork@draper.com)
- Fei Sun (fsun@draper.com)
- Michelle Beard (mbeard@draper.com)
- Robert Foley (rfoley@draper.com)
- Kyle Finley (kfinley@draper.com)
- Lewis John McGibbney (lewismc@apache.org)

Affiliations

- Draper
  - Joshua Poore (jpoore@draper.com)
  - Laura Mariano (lmariano@draper.com)
  - Clayton Gimenez (cgimenez@draper.com)
  - Alex Ford (aford@draper.com)
  - Steve York (syork@draper.com)
  - Fei Sun (fsun@draper.com)
  - Michelle Beard (mbeard@draper.com)
  - Robert Foley (rfoley@draper.com)
  - Kyle Finley (kfinley@draper.com)
- NASA JPL
  - Lewis John McGibbney (lewismc@apache.org)

Sponsors

Champion

- Lewis McGibbney (NASA/JPL)

Nominated Mentors

- Paul Ramirez (NASA/JPL)
- Lewis John McGibbney (NASA/JPL)
- Chris Mattmann (NASA/JPL)

Sponsoring Entity

The Apache Incubator

References

Mariano, L. J., Poore, J. C., Krum, D. M., Schwartz, J. L., Coskren, W. D., & Jones, E. M. (2015). Modeling Strategic Use of Human Computer Interfaces with Novel Hidden Markov Models. [Methods]. Frontiers in Psychology, 6. doi: 10.3389/fpsyg.2015.00919 Poore, J., Webb, A., Cunha, M., Mariano, L., Chapell, D., Coskren, M., & Schwartz, J. (2016). Operationalizing Engagement with Multimedia as User Coherence with Context. IEEE Transactions on Affective Computing, PP(99), 1-1. doi: 10.1109/taffc.2015.2512867