INSTITUTE FOR SOCIAL RESEARCH • SURVEY RESEARCH CENTER
SURVEY RESEARCH OPERATIONS
UNIVERSITY OF MICHIGAN

Capturing Survey Client-side Paradata

Hueichun Peng, Jason Ostergren

Survey Research Center

This Presentation

- ISR Client-side Paradata enhancements
  - Capture more events on mobile platform
  - More enhancements
  - Challenges

- Implementation on Blaise-5 surveys

Paradata

- Started with Web Survey Project
- Server-side Paradata
  - Question items, answers provided and time spent on each page
- Client-side Paradata
  - Respondents' keystrokes and mouse clicking behavior.
  - info of the respondents' browser/computer environment (OS, browser type, screen size, etc)

Client-side Paradata (CSP)

- Data of our 1st version CSP
  - ^t=1436:endScrollForCSP(x=false,y=true)
  - ^t=1780:Q2.CONSENT=true
  - ^t=1248:DATSTAT.NEXT=[CLICK]

  - ^t=1891:endScrollForCSP(x=false,y=true)
  - ^t=1732:FCST52_CENTRAL=16
  - ^t=1226: DATSTAT.PREV=[CLICK]

2016 CSP Enhancements

- Increased respondents on Mobile Platform
  - Around 20 ~ 25% among younger Respondent
- Events
  - Screen Size in Pixels and Device Size in Pixels
  - Tap (JavaScript Touch Events): equivalent to Click event
  - Orientation Change (Portrait or Landscape)
  - Pinch In/Out and Pinch distance

Future CSP Enhancements

- Geolocation
  - Need R's consent
- Coordinates of the object being clicked/touched on
  - Which (x,y)? Page, Screen, Viewport?
- Window focus/blurred
- Swipe
- Double Tapping

Implementing CSP in Blaise 5

- In general, 3 parts needed for capturing CSP
  - A JavaScript to capture CSP on the client browser
  - A mechanism to save the CSP during page change
  - A mechanism to update CSP with useful metadata identifiers (replacing temporary reused identifiers with fieldnames and code values)

Why is CSP capture needed in Blaise 5?

- Blaise 5 provides some CSP in out-of-the-box Audit Trail
  - Data entry events on page are individually recorded, not just the final answers on the page
  - This allows us to see what answers were selected or typed in what order, which may aid detection of problems with the clarity of the questions or tasks
- Some CSP we want is not included with Blaise 5
  - For example, we want to detect scrolling on the page to provide insight into any difficulties that page layout or screen size may have imposed
  - We need to be able to see when such scrolling happened relative to other events (i.e. did they change their answer after they scrolled?)
  - We are also experimenting with capturing touch CSP such as pinch-zoom for the same reasons as scroll CSP

JavaScript details: touch CSP

- CSP specific to mobile devices' touch capabilities
  - Detecting touch events such as pinch-zoom involves interpreting relative motion of two input signals against spatial and temporal tolerances, so implementations vary
  - Because this is potentially complex, we have used 3rd-party libraries to handle these events
  - For our Blaise 5 implementation, we are currently using hammer.js to detect pinch-zoom
  - Still various limitations: for example, we noticed touch disabled by default in Firefox running in Windows, preventing detection on a Surface device

JavaScript details: other dependencies

- Our CSP script still written in standard JavaScript
  - In the future, it might be wise to make use of the JQuery library for our CSP script
  - However, we have run into at least one conflict with the older version of JQuery that shipped with Blaise 5, so we have been uncertain about the optimal way to approach JQuery dependency
  - Blaise 5 uses the require.js framework to organize script handling, and we altered our script to use that framework (making sure multiple scripts can handle initial load events can be kind of tricky without some sort of framework or planning)

Router details: Data Entry API code

- The Data Entry API allows you to alter the functionality of the data collection components (running in a browser or native app)
  - The following details are based on current (pre 5.1) Data Entry API architecture for ASP
  - Blaise 5.1 changes will require rewriting of this code, so the details will differ in the future

Router details: implementation

- The router implementation for CSP involves adding code to IDataEntryControllerASP events
  - ControlFactory.OnCreateDataFieldInputControl
  - ControlFactory.OnCreateCategoryInputControl
  - BeforeExecuteActions
- The router also has code to generate a hidden textbox to store CSP until the page changes

Router details: useful identifiers

- Element IDs captured initially are not useful
  - We capture element IDs with most events in order to know which question item the event involves
  - Those IDs do not have any human readable meaning, and they are temporary, reused IDs so they cannot be resolved accurately once the page changes
- Our server-side code must, therefore, be able to translate those temporary IDs to something that will be useful later to analysts

Router details: matching category IDs

- Need to catalog associations of fieldnames and categories with temporary element IDs of radio buttons and checkboxes while html is being generated
  - ControlFactory.OnCreateCategoryInputControl provides an opportunity to do this with enumerated types

```
ICategory category = e.DataObject;
string fieldName = category.Field.Name;
if (category.Field.ValueType.DataType == StatNeth.Blaise.API.DataRecord.DataType.Set)
{
    fieldName += "-" + category.Code;
}
string clientID = definition.ID;
StoreClientIds(clientID, fieldName);
```

Router details: matching textbox IDs

- Need to catalog associations of fieldnames with temporary element IDs of text input boxes while html is being generated
  - ControlFactory.OnCreateDataFieldInputControl provides an opportunity to do this with text entry types

```
string fieldName = field.Name;
string clientID = "";
string definitionID = definition.ID;
if (!string.IsNullOrEmpty(rawMask))
{
    clientID = definitionID + "_mask"; //This is for a UM custom control
}
else
{
    clientID = definitionID + "_tb";
}
StoreClientIds(clientID, fieldName); //Retains matches until page change
```

Router details: updating IDs

- Need to update CSP before storing it away by replacing temporary IDs with useful ones
  - BeforeExecuteActions provides an opportunity to do this

```
        if (_hiddenField != null)
        {
            IRouteItem paraDataField = _depcontroller.Page.RouteItems.GetItem("ClientSideParadata"); //This is a Field Reference
            if (paraDataField != null)
            {
                string csp = _hiddenField.Value; //This is a hidden text box we add to the html
                foreach (KeyValuePair<string, string> clientIdentifier in _clientIdentifiers)
                {
                    csp = csp.Replace(":" + clientIdentifier.Key, ":" + clientIdentifier.Value);
                }
                paraDataField.Value.Assign(csp);
                _clientIdentifiers = new Dictionary<string, string>();
            }
        }
    }
```

Blaise 5 code

- We store CSP in a Open field as a concatenated string which can be extracted normally
  - Concatenation is handled near the beginning of our Blaise 5 instrument source code itself

```
ClientSideParadata.keep
ClientSideParadataStore.keep
IF ClientSideParadata <> EMPTY THEN
   ClientSideParadataStore := ClientSideParadataStore + ClientSideParadata
   ClientSideParadata := EMPTY
ENDIF
```

  - The two Open fields used here are named the same for all our instruments; “ClientSideParadata” is the Field Reference used by the API

CSP Data Example

^t=5:onload-SecA.StartInterview.A006_=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:16.609Z

^t=2396:SecA.StartInterview.A006_=1[ENTR]

^t=0:onload-SecA.StartInterview.A007TRAlive_A=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:20.363Z

^t=3293:PinchApart(Scale=3.7391,Duration=1003)

^t=73:Scroll(x=true,y=true)

^t=2934:SecA.StartInterview.A007TRAlive_A=1

^t=1601:PinchTogether(Scale=0.2272,Duration=474)

^t=1238:Scroll(x=true,y=true)

^t=-124:SecA.StartInterview.A007TRAlive_A=[ENTR]

^t=0:onload-SecA.StartInterview.A002_IwBegin=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:30.211Z

^t=2884:SecA.StartInterview.A002_IwBegin=1[ENTR]

^t=2:onload-SecA.StartInterview.A155_SelfPrxy=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:34.123Z

^t=6062:Help=[CLICK]

^t=2581:SecA.StartInterview.A155_SelfPrxy=1[ENTR]

Question and Discussions

Contact Information

– Hueichun Peng hupeng@umich.edu

– Jason Ostergren jostergr@umich.edu