

Model Documentation

EM Assignment

Date: January 2013
By: Judy Palimonka (1266015)
Model Name: The Process of Adapting Adverts in Targeted Advertising
Weighting: 60 : 40

1. Purpose

The main objective of this model was to use the EM principles to describe the processes and concepts involved in Targeted Advertising. The knowledge gained while interacting with the model should help the user understand the privacy issues and make their own opinion with regards to it.

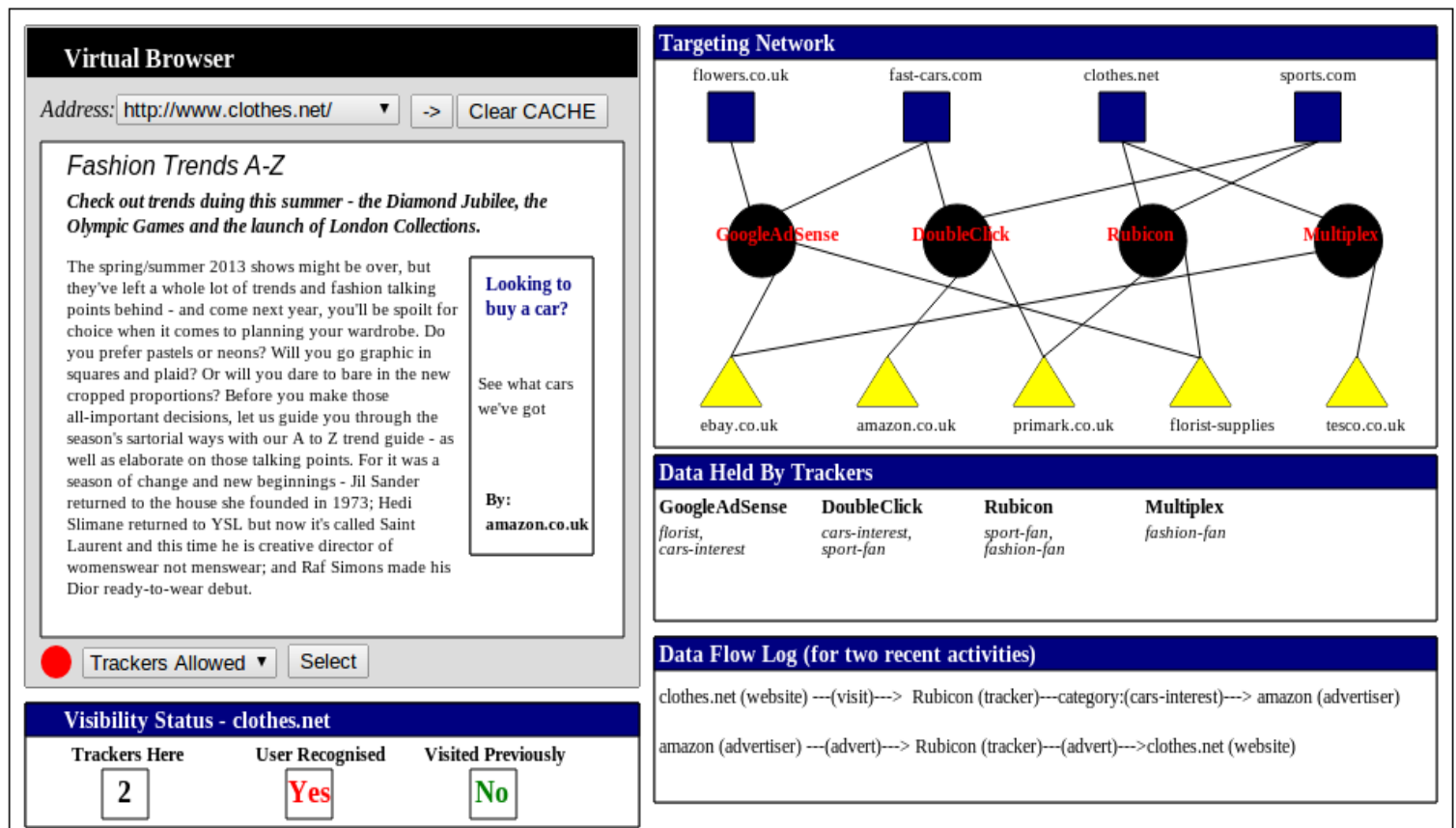


Fig1 Screenshot of TA model in JS-EDEN

2. Implementation and Compatibility

The model has been implemented using JS-Eden and is confirmed to be compatible the "latest" and "Emile" versions. Unfortunately I have been unable to make it compatible with the recently released version ("master") and I do not believe the model is at fault. This version does not recognise the references to canvas object.

3. Construal Objectives

While developing a construal by the means of interacting with the model, the modeller is expected to understand the following concepts and patterns:

- Targeted advertising is facilitated by web tracking
- The parties involved in web tracking are: website and tracker

- A website passes user data to a tracker or a number of trackers
- A tracker can be a partner with many websites
- A website can be a partner with many trackers
- User may be recognised by a website even though he/she hasn't been to the website
- If tracking is blocked, the number of trackers does not increase and there are no targeted adverts
- Targeted advertising can be prevented by blocking trackers
- The collected by trackers data can be removed by clearing the CACHE
- Targeted adverts reflect on previously visited websites

4. Structure

This part discusses the EM concepts used in the model. I also included the key procedures as well as JavaScript functions.

4.1 Agents

Modeller is a super-agent and is directly or indirectly responsible by triggering all changes in the state of model's observables. We also recognise agents that cause random changes in the model, i.e. **Targeted Advert Generator** defines the correlation between the number of observables and dependencies that assist in generating targeted adverts. **Party Detector** identifies the parties that take part in targeted advertising process and the relationships between them. Then it passes the information onto Targeted Advert Generator.

4.2 Explicit Observables

For the purpose of this model documentation, I will refer to the visible parts of this EM artefact as explicit observables. This review is more comprehensive than the one included in the paper. Each of the observable is connected to more than one variable/constant/observable contained within JS/definitive scripts. Some observables are visualised in the graphical representation of construal e.g. the number of trackers on site correspond to the box in visibility status. They represent obvious dependencies between the state of the model and its graphical representation in the boxes neighbouring the virtual browser. The list of explicit observables is as follows:

- website
- tracker
- segment
- advert
- advertiser
- the state of blocking (corresponds to value of the cmbBlocking combo box)
- targeting network
- data held by trackers
- data flow log
- visibility status – user recognised
- number of trackers present on web page
- currently selected website (corresponds to value of the cmbSites combo box)
- website previously visited
- virtual browser

4.3 Implicit Observables

Those observables that help update or represent the state of the previously explicit variables are shown in Figure 1. All denoted with “TA” before the name (the names in model however do not have prefix TA), the names have been changed to focus all the observables related to this model in one location.

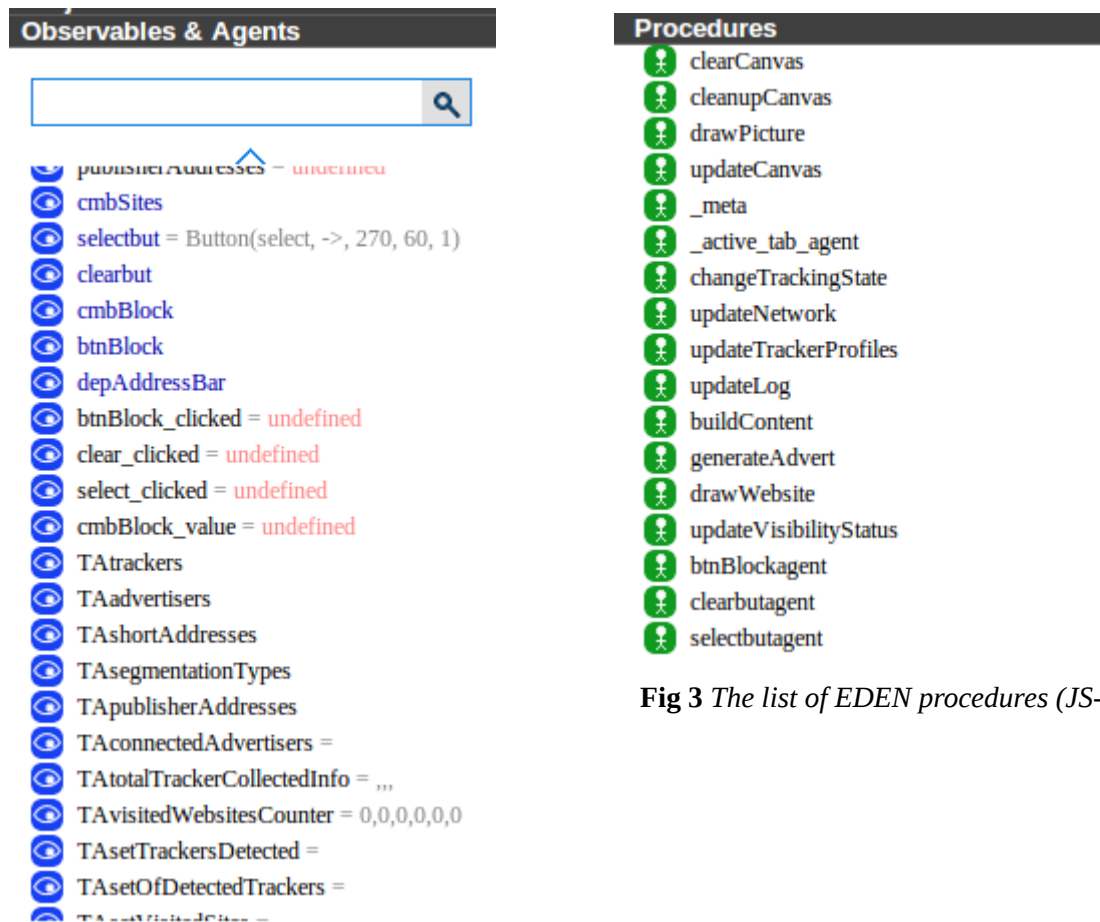


Fig 3 The list of EDEN procedures (JS-EDEN)

Fig 2 The list of observables (JS-EDEN). Most of them being both JS variables and EDEN observables

4.4 Dependencies

Expressed as arrays and also can be seen in the advertising network. The arrays below (apart from the first one) form one-to-many relationships in such a way that the the id of first concept is expressed as the array position and the many items that are connected to it, are array entries. .

```
depAddressBar is [cmbSites,selectbut,clearbut,cmbBlock,btnBlock]
seenAdverts=[ [],[],[],[],[],[] ] updated dynamically
siteTrackerLinks = [ [0],[0,1],[2,3],[1,2],[2,3],[3] ]
websiteSegmentLinks = [[0],[1],[2],[3],[4],[5]]
trackerAdLinks = [[0,3],[1,2],[2,3],[0,4]]
segmentAdvertiserLinks = [ [3],[0],[1,2],[1],[4],[0,1] ]
```

4.5 Constants

The following include the values that never change, e.g. the addresses for website, content, advert headers, etc. Also TA is a JS objects used to translate a few JS variables into EDEN.

```
trackers
advertisers
shortAddresses
shortAdvertisers
publisherAddresses
segmentationTypes
advertHeaders
advertContent
contentHeaders
```

subtitles (for adverts)

siteContent

TA (an object holding all JS variables that are translated into EDEN)

4.6 Procedures

The following procedures are triggered on super-agent's command to update the website, clear/disable/enable the tracking. The names are self-explanatory – every procedure updates the corresponding part of the model. All, apart from the last procedure are run in the order as designated below. ChangeTrackingState() obviously triggers when the modeller uses the combo box button to block trackers. The two two procedures that are triggered at the mouse click have not been included.

buildContent()

drawWebsite()

updateNetwork()

updateTrackerProfiles()

updateLog()

generateAdvert()

updateVisibilityStatus()

changeTrackingState()

4.7 JavaScript Functions

decl(name,value) - declare a JS variable as an EDEN observable

updateAll (array) – make the observables equal to the JS variables

getVal(name) - get value of an observable

clearMemory() - reset the value of certain observables and change the state to the initial value (usually an empty array or 0)

wrapText(text,type,xx,yy) - write text on canvas within the area specified

findAdvert() - generate a random advert accordingly to the pages user has viewed so far

showData(tracker,advertiser,website) - show data in the log

5. Model Manual

1. Pick a website using the combo box and observe how it changes the neighbouring sections of the model
 - Check the Visibility Status box
 - Observe how new advertisers and trackers have been added to the network. Note that one tracker can be connected to one or more website. Therefore even when you go to website you haven't visited before, if the tracker on the network is associated with this website, you will be recognised. (indicated by “User Recognised” in Status Visibility)
 - Observe what data the trackers have gathered so far by examining “Data Held By Trackers”. You should be able to recognise it's only a customer category rather than personal information
 - Inspect the content of advert, its origin and the subject of offer
 - Examine how the content of advert is related to the data flow included in Data Flow Log
2. Change the state of the entire model by selecting Trackers Blocked from the combo box underneath the website content
 - Choose to browse websites using the combo box in Virtual Browser
 - Observe the impact of blocking on advertising network and data flow.
3. Change back to Trackers Allowed. Browse again then click “Clear CACHE” to observe the effects of clearing browsing history.

6. System Manual

Go to <http://jseden.dcs.warwick.ac.uk/emile/> and simply paste the contents of TA_model script into the EDEN interpreter window