## Introduction to tool

You can use JSpoor to record user interface events which happen during users working sessions with a java/Swing GUI application. It captures both low level events like mouse moves and semantic (or high level) events like menu item selections. Also, you are able to extract statistical data from usage logs. This software uses AspectJ for logging swing components events. It uses AspectJ's Load Time Weaving.

## Install & use

In order to use this tool, you need AspectJ installed on your machine. You can download AspectJ and its source code via http://www.eclipse.org/aspectj/. AspectJ is developed under Eclipse Public License - v 1.0. aj5 tool location should be on PATH. AspectJ Jars such as aspectJrt.jar, aspectjweaver and etc. should be on CLASSPATH or at root directory of JSpoor.
To log users' interactions with a java/swing application:

1. Download Jspoor package (it contains jspoor.jar in its dist directory). You can also build it from source by running the ant build file.
2. Then you need to modify the "run.bat" batch file by adding your applications jar file location to its classpath (Instead of sample terppaint.jar). The location of jspoor.jar and aspectj.jar should be set there also. Also, the last line should be modified and the name of the main class of your java application should be inserted at the end of line (instead of paint.TerpPaint). Now, execute this batch file. The output log file will be created at the same directory where "jspoor.jar" is located.

## What will be logged to files?

At output log file many attributes related to the recorded event is logged also. Usually, the first column is event name, second column is the UI component name originating that event, third column is the time when that event has happened (time passed since the user has started interacting with UI). below are some parts pf jspoor's output file while a user uses a java paint software (TerpPaint available https://sourceforge.net/projects/terppaint/):

```
MOUSE_DRAGGED: TerpPaint: 15187: (71,71)
MOUSE_DRAGGED: TerpPaint: 15297: (91,80)
MOUSE_DRAGGED: TerpPaint: 15297: (101,84)
MOUSE_DRAGGED: TerpPaint: 15390: (183,91)
MOUSE_DRAGGED: TerpPaint: 15422: (214,88)
TOOLTIP_SHOWED: Fill With Color: 6016
MOUSE_MOVED: TerpPaint: 6422: (36,83)
MOUSE_MOVED: TerpPaint: 6438: (36,85)
TOOLTIP_HIDED: null: 6563
TOOLTIP_SHOWED: Magnifier: 6578
MOUSE_MOVED: TerpPaint: 7516: (39,123)
MOUSE_MOVED: TerpPaint: 7532: (39,125)
TOOLTIP_HIDED: null: 7532
TOOLTIP_SHOWED: Brush: 7547
MOUSE_BUTTON_PRESSED: BUTTON1: 18610
TOOLTIP_HIDED: Brush: 18610
MOUSE_BUTTON_RELEASED: BUTTON1: 18797
BUTTON_PRESSED: Brush: 18797
MOUSE_MOVED: TerpPaint: 20297: (40,133)
COMPONENT_FOCUS_GAINED: null: 37250: JTextField
MOUSE_MOVED: Text: 37469: (29,50)
MOUSE_MOVED: Text: 37469: (26,52)
MOUSE_MOVED: Text: 37469: (20,54)
MOUSE_MOVED: Text: 37500: (10,56)
KEY_PRESSED: H + Shift: 39531
KEY_PRESSED: E: 40094
KEY_PRESSED: L: 40547
KEY_PRESSED: L: 40656
KEY_PRESSED: L: 41094
KEY_PRESSED: O: 41328
KEY_PRESSED: Backspace: 49609
MOUSE_BUTTON_PRESSED: BUTTON1: 54765
COMPONENT_FOCUS_LOST: null: 54875: JTextField
MOUSE_MOVED: TerpPaint: 68953: (95,246)
MOUSE_MOVED: TerpPaint: 69062: (95,245)
MOUSE_BUTTON_PRESSED: BUTTON1: 69172
MOUSE_BUTTON_RELEASED: BUTTON1: 69250
MENU_ITEM_SELECTED: Exit: 69312
```

Different parts of messages logged for each type of event is described at Table 1.These parts are separated via a ‘:’ character.

**Table 1 - message parts and their descriptions**

| Event type | First parameter | Second parameter | Third parameter |
|---|---|---|---|
| MENU_ITEM_SELECTED | event source name | time | |
| POPUP_MENU_SHOWED | event source name | time | |
| MENU_ITEM_NAVIGATED | event source name | time | |
| BUTTON_PRESSED | event source name | time | |
| RADIOBUTTON_SELECTED | event source name | time | Is selected or deselected |
| CHECKBOX_SELECTED | event source name | time | Is selected or deselected |
| COMPONENT_FOCUS_GAINED | event source name | time | event source component type |
| COMPONENT_FOCUS_LOST | event source name | time | event source component type |
| JSLIDER_CHANGED | event source name | time | JSlider value selected |
| COMBOBOX_SELECTED | event source name +"selected" | time | Selected item |
| TOOLTIP_SHOWED | The tip showed | time | |
| TOOLTIP_HIDED | The tip showed | time | |
| MOUSE_BUTTON_PRESSED | Button type | time | |
| MOUSE_BUTTON_RELEASED | Button type | time | |
| KEY_PRESSED | Pressed key name plus modifiers if available | | |
| MOUSE_MOVED | event source component type | time | (X,Y) The mouse cursor location on the screen |
| MOUSE_DRAGGED | event source component type | time | (X,Y) The mouse cursor location on the screen |
| ADJUSTMENT_VALUE_CHANGED | event source component type | time | |

## Extracting statistics and attributes from log files

You can extract values for some attributes related to usres' performance from log files recorded by jspoor. You can access this tool both programmatically and from command line.
To extract attribute values programmatically, jspoor jar file should be on your classpath. This tool needs some files to be in its root directory before being able to operate, these files are: attrsList.txt, noises.txt and shelldata.txt.
Noises.txt file contains noisy steps at each step. Each row in this file starts with trial number, then a space and then a comma separated list of noisy step numbers in that trial, for example:

*1 5,6*
*6 2*
*10 3*
*13 2*

attrsList.txt contains the name of attributes which we 'd like to be included in the generated report and their values will be calculated, this is used in generating some types of reports such as arffs. for example:

*stepTime*
*avgItemsnavigatediInEachSession*
*avgMenuNavigationTime*
*avgMenuItemNavigataionTime*
*avgComboSelectionTime*
*avgVelocityBeforeButtonClicks*
*menuNavigationsCount*
*avgOfAllPauses*
*avgKeyTypedPause*
*distanceTravelledSum*
*avgMouseXVelocity*
*avgMouseYVelocity*

shelldata.txt file contains data about the task which is performed by users. It has a general structure like this:

*numOfSteps*
*actionsCountInSteps*
*NOVICE_NOVICE_HIGH,NOVICE_MODERATE_LOW,NOVICE_MODERATE_HIGH,NOVICE_SKILLED_LOW*

*EXPERT_NOVICE_HIGH,EXPERT_MODERATE_LOW,EXPERT_MODERATE_HIGH,EXPERT_SKILLED_LOW*
*MIN_PAUSE_DURATION*
*step1KlmTime,...,stepNKlmTime*
*minStep1Distance,...,minStepNDistance*
*numOfResponseTimeActions*
*RespTimeID, responseTime, condition*
.
.
*step1action1,...,step1actionN*
.
.
*stepNaction1,...,stepNactionN*
*Step1PerformanceIndicator*
.
.
*StepNPerformanceIndicator*
*startActionIndictor*

to generate weka arff files for a log file from your java codes, add code lines like the following:

```
ExtractVariableValuesFromUILogs instance = new ExtractVariableValuesFromUILogs();
instance.extractAttributeValues("stepArffAS", true,
                                        "C:\\columbauser\\ui-log_Sat_11-Apr-2009_16-36-39.txt",
                                        "skilled", 10, true,  false);
```

The method signature is as following:

```
/**
         * generates the specified report from the given ui log file
         *
         * @param reportType
         *            report type
         * @param normalizd
         *            calculate normalized values or task-dependent values
         * @param logFilePath
         *            the path to UI actions log file
         * @param skillLevel
         *            user's general or system skill level
         * @param trialNo
         *            performance trial number
         * @param dontCheck
         *            if set to true then the sequence of actions done in log file
         *            is not compared and cheked against the sequence indicated in
         *            shelldata file
         * @param areOldLogs
         *            this should be set to true for log files generated by older versions
         *              of jspoor in which the x, y position of mouse in each dialog is
         *  recorded according to its top left position(0,0)
         */
public void extractAttributeValues(String reportType, boolean normalizd,String
logFilePath, String skillLevel, int trialNo, boolean dontCheck,boolean areOldLogs)
```

if MODERATE_HIGH values are set to negative values (in shelldata file) then only two levels of skills(novice and skilled) are considered. In the command line mode (or batch mode), you can generate reports for a task performance repetitions for some users just with running one command. In order to use this tool from command line, type the following command:

*Java –jar jspoor.jar  reportType logDirsList -normalized*

reportType indicates the type of report. reportTypes and their descriptions are explained in Table 2. If normalized option is available then task-independent values are calculated for attributes, otherwise tasked-pendant values are calculated. logDirsList is a text file in which each line is the path to a directory of log files, for example:

*C:\user1*
*C:\user2*

It is assumed that each of these directories contain a sub-directory called *logs* in which there are a number of UI log files. These UI log files are resulted from repetitive performance of a task by a user, it is assumed that file names alphabetical ascending sorting indicates their ordering (first trial to last trial). Also in of these directories two text files should be available. The first one is noises.txt, which was described earlier and is used to specify which steps in trials are noisy and should be removed from calculations. The second file is user-info.txt. This file contains the three following lines:

*User-ID*
*Gender(m or f)*
*System (general) skill level (skilled or novice)*
*absolutePos or relativePos (this is for compatibility with UI log files created by older versions of this tool, for new versions use absolutePos value)*

The gender value is not used in the current version of the jspoor, so you can assign any value to this attribute. Mainly, there are two types of reports. One group is weka arff reports which are

generated at the root directory of jspoor. The second group is attribute-across-trials reports. In these reports one or a few attributes are extracted for each step and in all trials of users. For example, for a 7 step task, 7 files will be generated at a directory which has the name of reportType at each user's directory (directories specified at logDirsList). In each of these files, values for related to that report type will be printed in front of the trial numbers (each line shows values for one trial).

**Table 2 - report types**

| Report Name | Description |
|---|---|
| stepsTime | Steps performance time |
| klmRatio | Steps time divided by their KLM-GOMS prediction |
| velocityInfo2 | Prints attribute values related to mouse movements.<br>The attributes printed and their ordering is as follows:<br>episodesCount,distanceTravelledSum,directionChangesCountSum, totalAngelChangesSum , moveDurationSum,avgXVelocity , avgYVelocity,avgVelocity, avgXAccelerration, avgYAcceleration, avgAcceleration, maxXVelocity, maxYVelocity , maxVelocity, maxXAccel, maxYAccel,maxAccel, avgVelocityBeforeButtonClicks |
| comboVelocityInfo2 | Prints attributes values related to mouse movements during combo box selections (the same attributes as velocityInfo2). |
| dragVelocityInfo2 | Prints attributes values related to mouse movements during drag operations (the same attributes as velocityInfo2). |
| comboTimes | Combobox selection times |
| menuReport | Prints attributes related to interacting with menus. These attributes and their order is as following:<br>avgItemsnavigatediInEachSession,avgMenuNavigationTime, avgMenuItemNavigataionTime , menuNavigationsCount ,<br>, avgSelectedMenuItemsDwellTime , avgVibrationRadius<br>, avgVibrationCount |
| responseTimeBehavior | Prints the attribute value related to users adaptation to response times |
| minDistanceReport | Prints the minimum distance which The mouse cursor needs to travel in order to perform each step |
| purePauses | Pauses users do after the response times end |
| typePause | Pause between typing two letters on keyboard |
| tooltipReport | Prints attribute values related to tooltip viewing.<br>These attributes and their order is as following:<br>toolipViewedCount,tooltipViewingSessions, avgTooltipsViewedCountInSession, |

| | |
|---|---|
| | avgTooltipViewTime<br>,avgTooltipSessionTime |
| pausesReport | Prints attributes related to pause times.<br>pauseCountS, avgPauseTime, beforeStepPause, avgActionPause, avgAllPauses |
| stepArffAS | Generates weka arff files which are labeled by users' application skill levels |
| stepArffGSAS | Generates weka arff files which are labeled by both users' application skill levels and system skill levels |
| stepArffGS | Generates weka arff files which are labeled by users' System skill levels |
| stepArffASDif | Generates weka arff files which are labeled by users' application skill levels. The attribute values are the subtraction of two succeeding trials |
| stepArffGSASDif | Generates weka arff files which are labeled by users' system skill levels and application skill levels. The attribute values are the subtraction of values in two succeeding trials. |
| stepArffGSDif | Generates weka arff files which are labeled by users' System skill levels. The attribute values are the subtraction of values in two succeeding trials. |
| stepArffGSasParamAS | Generates weka arff files which are labeled by users' application skill levels and users' system skill are considerd as an input attribute. |