# Development of Log Files Analysing Tool for Bmc Remedy Ar System

Lavanya G S[1], and Prof Pushpa H G[2]
[1]M.tech Student, Department Of Computer Science, SBMJCE, Bangalore, India
[2]Head of the Department, Department Of Computer Science, SBMJCE, Bangalore, India

***Abstract*—Log files contain large amount of data which is used while debugging an application. Analyzing these software log files helps during testing and troubleshooting. A log file analyzer for BMC Remedy Action Request System (AR System) is presented in this paper. Log files grow very quickly depending on user activity and type of logging. Many times these log files grow to 200-300 Mbytes within 4-5 minutes. To physically go through such huge log files is very painful and consumes lot of time as well. To resolve this, application that can parse through these log files and trace the execution of workflows for given user and present the same in a graphical & intuitive interface is presented.**

***Key words*—Log files,AR System, workflow, graphical interface**

## I. INTRODUCTION

The logging activity is mainly been used as a debugging aid which is particularly vital to distributed applications. Large software systems often keep log files of events. Such log files can be analyzed to check whether a run of a program reveals faults in the system. The runtime information that is being recorded in the log files provides an easy and very powerful debugging approach. Pausing program execution at a breakpoint disturbs the time order of interacting with subsystems as the software runs in a controlled environment [1]. Hence this logging activity sometimes turns out to be the most feasible debugging method. In automated log file analysis data from the log files is extracted. This requires decoding the log file syntax and interpreting data semantics. The output of this phase is used by an organization for further processing. The log data extractors can be developed using programming languages targeting log file formats [2]. A generic scheme for interpreting elements of a log file is desired instead of repeating this process for each log file format and using suitable data structure for further processing. Almost every tool present in the market generates some form of a log which can be used to identify software defects that can be hardly detected in conventional testing.

AR System is a professional development environment that leverages the best practices of the IT Infrastructure Library (ITIL) and provides a foundation for Business Service Management (BSM) solutions. In addition, log file analysis also helps in extracting vital information from AR servers and in security monitoring. The output of the AR System log analyzer is a tree filled with the information of interest for the particular case. The log files generated in AR server are analyzed for many purposes. In particular, the goal is to verify the functional conformance of the application with a given specification for each workflow in the AR System.The log file entries are observed to confirm that the application generates desired outputs at intended instances. When an application malfunctions in production, the only trace available for developers is to investigate the cause, which more often, leads to the examination of the application log file.

Powerful log files analysis tools are required with capabilities to monitor the execution of different workflows, the order of execution, number of users connected to server, timestamp related to each workflow object etc.for AR System. Correlating data extracted from a monitoring tool with an application log can reveal valuable information caused by important application events. Despite the benefits of it, log file analysis is a labor intensive and error prone activity when performed manually. Furthermore, it generally demands domain and tool expertise which is an expensive resource. Therefore, automating at least a part of the process has significant importance. The first step in automated log analysis is automatic extraction ofrelevant information from log files. Doing this in a generic way is a challenging task given that different log files generated in the AR server have different structures and formats.

## II. RELATED WORK

Log files have great importance in AR System since they help us in identifying the root-cause of an incident. If thereis a production release, logs might be full of errors and these errors must be ordered according to their impact. These errors are then solved one by one till a clean log file is obtained. AR LogAnalyzer is an existing program that is used to analyze AR System SQL and API logs only. The output provides a breakdown of execution times for each API call and SQL statement and, for API logs, a breakdown of idle times by thread. Collectively, this output is used to pinpoint performance trouble spots and to help guide AR System Administrators generally in their efforts to do performance tuning.

This tool requires a command-line script with no built-in GUI interface or prompting. It does not come with specific guarantees concerning the operation of the utility or the accuracy of its output. AR LogAnalyzer does not provide any graphical representation for the given log files; neither does it provide color codes for easy debugging. As this tool is operational from the command prompt it is not user friendly. To overcome these draw back, log analyzer tool is developed which provides reliable results along with tree structure representation of the contents of the log files.

## III. SYSTEM ARCHITECTURE

The main components of an AR System application include:

- Forms - The main AR System application component that users interact with is a form. Each form is composed of fields.
- Menu—Menus are lists that you create to guide the user in entering information in fields on forms.
- Workflow—Forms provide the mechanism to structure the data captured and menus offer options for specific field data, additional components—active links , filters, and escalations—act on the data to automate business processes, or workflow. These components trigger actions in response to execution options that is defined.
  - An active link is an action or group of actions performed on the client. Active links are triggered by user actions in a form.
  - A filter is an action or group of actions performed on the AR System server. Filters are used to enforce business rules and to ensure system and data integrity.
  - An escalation is an action or group of actions performed on the server at specified times or time intervals.

The main components of the parser include user interface, AR server, log parser and finally the graphical view of the logs processed.

AR Server: A log file records everything that goes in and out of a particular server. The information is frequently recorded chronologically, and is located in the root directory, or occasionally in a secondary folder, depending on how it is set up with the server. The only person who has regular access to the log files of a server is the server administrator. A log file is generally password protected, so that the server administrator has a record of everyone and everything that wants to look at the log files for a specific server. Since these servers are built with proprietary technologies, only way to debug any application issues is to go through the log files generated and find out which workflow fired/not fired. Problem with this approach is that these log files grow very quickly depending on user activity and type of logging. Also, since these are server side logs, they contain workflows executed for all users and not just the user who is having issues. To physically go through such huge log files is very painful and consumes lot of time as well. In the process productivity of developer also gets affected.

Log Parser: Log file parser is needed that can parse through these log files and trace the execution of workflows for given user and present the same in a graphical & intuitive interface. This would greatly reduce time & effort spent by developers in troubleshooting application issues.

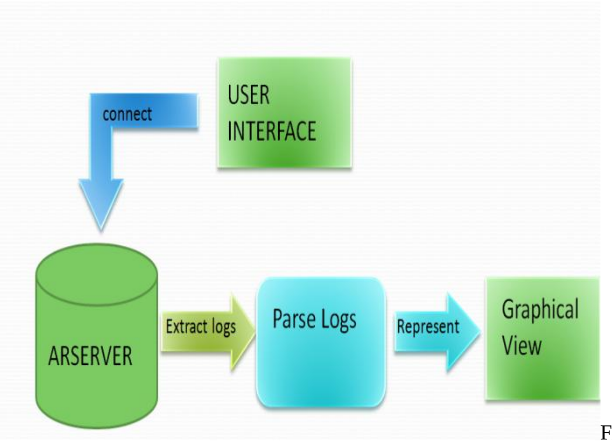


Figure 1: System Architecture

User Interface: User Interface module allows user to connect any AR Server located in any geographical area of an organization. User should know the AR Server name to login into server. Once user logs using valid credentials into server, he is able to give particular log file name to parse and to get all information related to the log file. User request the system through User Interface. User request provides the transaction id in particular field.

Extract and parse: In this module program extracts particular log file mentioned by user and give it to parsing. During parsing, log file will be read line by line and separates active links and filters present into two different text files. Based on the one of the conditions, i.e., transaction id (TID) or user, parser will execute and provide a graphical tree representation of the same. Here we are using pattern matching as our key concept to separate active links and filters and also to find which workflows triggered and which did not.

Graphical View: The execution of workflows is presented in a tree structure which helps in easy debugging of the application. User is Capable of viewing workflow based on particular user or TID assigned to a user. The tree structure also indicates whether selected workflow is executed or not, whether selected workflow is enabled or disabled and the actions performed if this workflow is executed.

## IV. IMPLEMENTATION

The function of workflow is to process the data captured in forms in accordance with the business needs. In AR System, workflow automates a company's processes through the use of active links, filters, and escalations. In general, workflow can be defined as the set of processes that a company uses to run itself. Each of these actions will be recorded in the log files.

The application developed can parse through these log files and trace the execution of workflows for given user and present the same in a graphical & intuitive interface. This would greatly reduce time & effort spent by developers in troubleshooting application issues. The application developed has the capabilities:

- Capable of parsing proprietary log files generated by BMC Remedy Applications
- Capable of parsing log files up to 1 GB of size.
- Capable of identifying workflow execution per user.
- Display execution of workflows in a graphical & intuitive manner.
- It has the ability to highlight workflows on User Interface(UI) and shows following details about the workflow when highlighted:
    - Whether selected workflow is executed or not.
    - Whether selected workflow is enabled or disabled.
    - Workflow execution condition.
    - Value of parameters set while executing this condition.
    - Actions performed if this workflow is executed.

The proposed system working architecture is as shown in figure 2. Here the user extracts the logs from BMC AR server by connecting to it, the extracted log files are then parsed and represented in a form which is useful to the developer in troubleshooting the issues.

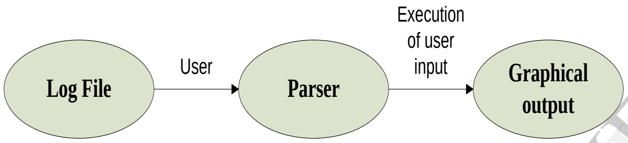


Figure 2: Proposed System

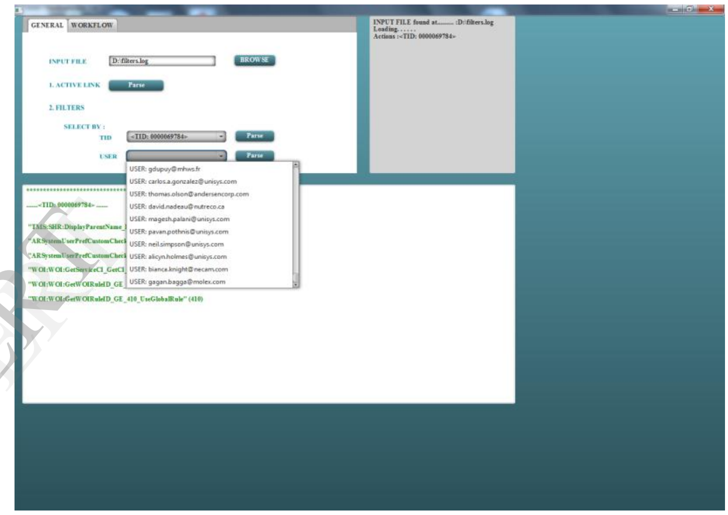
The central servers in an organization will be having huge amount of log files in terms of GB or TB those get generated every day. Our program will get access to AR Server only when particular server is accessed by authenticated user. The program will search for particular log in AR server database. If found it retrieves file name and load into the parser for parsing a log file. If it is not found then display error message to the user on UI.

The Parser is responsible for extracting the log file from AR server database into parsing tool. It separates remedy objects; active links, filters and other workflows. To separate the remedy objects, pattern matching using regular expressions approach is used. The log files are also processed using different condition based on workflow names that have triggered, users logged on to the server and also the transaction id used by particular users. The tool also provides information regarding the number of users currently connected to the server and their last modification time. For a given workflow present in the log file the parser returns a detailed list of actions performed, the workflow objects primary form name, the execution order of the workflow etc.

Graphical View: is responsible for viewing all workflows according to the user triggering the actions. User can view the workflows related to only active links or filters. Users can also view the workflow of different users to trace the execution of each workflow present in the server. This view is designed like tree structure with an expandable option which when clicked reveals the actions performed if any. Each active link/filter name is colored green or red based on the passed or failed qualification. All the disabled workflows are represented in blue. Each specific action is represented by unique color in the tree structure. Thus, by just observing the graphical representation developers can easily identify the cause of error based on the qualifications used to trigger each workflow. Totally this gives complete representation of what user has done from his system and the course of actions that has followed during the transactions.

## V. RESULTS

The main page of the tool is as shown in figure 3. The user provides the log file generated by BMC AR Server. The log file is selected by clicking on browse button through a file chooser. Once the file is selected user selects the criteria to parse the file as shown in figure 3.



```
INPUT FILE found at......... :D:\filters.log
Loading. . . . . .
Actions :<TID: 0000069784>
```

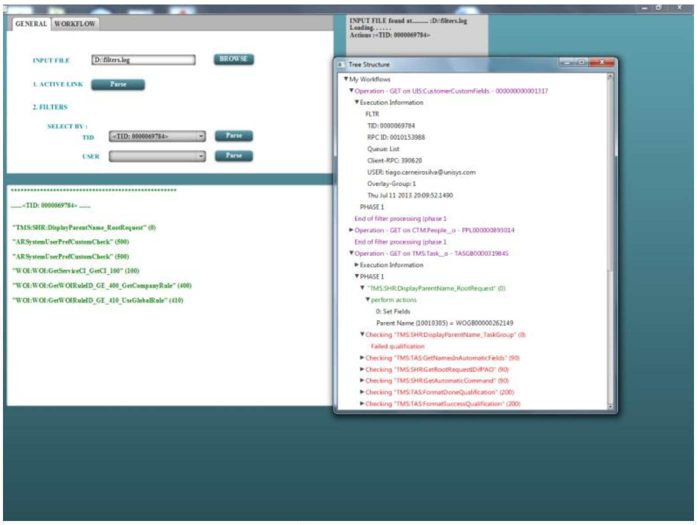
```
USER: gdupuy@mhws.fr
USER: carlos.a.gonzalez@unisys.com
USER: thomas.olson@andersencorp.com
USER: david.nadeau@nutreco.ca
USER: magesh.palani@unisys.com
USER: pavan.pothnis@unisys.com
USER: neil.simpson@unisys.com
USER: alicyn.holmes@unisys.com
USER: bianca.knight@necam.com
USER: gagan.bagga@molex.com
```

```
********************************
.......<TID: 0000069784> .......

"TMS:SHR:DisplayParentName_]
"ARSystemUserPrefCustomCheck
"ARSystemUserPrefCustomCheck
"WOI:WOI:GetServiceCI_GetCI_
"WOI:WOI:GetWOIRuleID_GE_
"WOI:WOI:GetWOIRuleID_GE_410_UseGlobalRule" (410)
```

Figure 3: Selection of criteria to parse the log file

The parsing of log file is done based on Thread ID and also based on Users present in the log file. As seen in Figure 4, parsed graphical representation of log file based on Thread ID is done.



```
INPUT FILE found at......... :D:\filters.log
Loading. . . . . .
Actions :<TID: 0000069784>
```

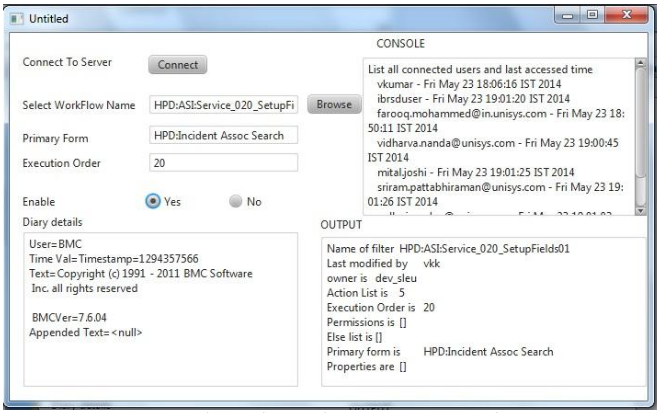
```
*****************************************************
.......<TID: 0000069784> .......

"TMS:SHR:DisplayParentName_RootRequest" (0)
"ARSystemUserPrefCustomCheck" (500)
"ARSystemUserPrefCustomCheck" (500)
"WOI:WOI:GetServiceCI_GetCI_100" (100)
"WOI:WOI:GetWOIRuleID_GE_400_GetCompanyRule" (400)
"WOI:WOI:GetWOIRuleID_GE_410_UseGlobalRule" (410)
```

```
▼My Workflows
 ▼Operation - GET on UIS:CustomerCustomFields - 000000000001317
  ▼Execution Information
    FLTR
    TID: 0000069784
    RPC ID: 0010153988
    Queue: List
    Client-RPC: 390620
    USER: tiago.carneirosilva@unisys.com
    Overlay-Group: 1
    Thu Jul 11 2013 20:09:52.1490
   PHASE 1
  End of filter processing (phase 1
 ►Operation - GET on CTM:People_o - PPL000000893014
  End of filter processing (phase 1
 ▼Operation - GET on TMS:Task_o - TASGB0000319845
  ►Execution Information
  ▼PHASE 1
   ▼ "TMS:SHR:DisplayParentName_RootRequest" (0)
    ▼perform actions
      0: Set Fields
      Parent Name (10010305) = WOGB00000262149
   ▼Checking "TMS:SHR:DisplayParentName_TaskGroup" (0)
     Failed qualification
   ►Checking "TMS:TAS:GetNamesInAutomaticFields" (90)
   ►Checking "TMS:SHR:GetRootRequestIDifPAO" (90)
   ►Checking "TMS:SHR:GetAutomaticCommand" (90)
   ►Checking "TMS:TAS:FormatDoneQualification" (200)
   ►Checking "TMS:TAS:FormatSuccessQualification" (200)
```

Figure 4: Generation of tree structure based on the TID

By selecting the user text box, the workflow for that particular user can be represented graphically in similar fashion as shown in figure 4. For each selection the names of the forms that contains the workflow names is displayed in the text box area.  A console area is also provided where the background action performed is listed as shown in figure 5.



Connect To Server

Select WorkFlow Name: HPD:ASI:Service_020_SetupFi

Primary Form: HPD:Incident Assoc Search

Execution Order: 20

Enable: ● Yes ○ No

Diary details

```
User=BMC
Time Val=Timestamp=1294357566
Text=Copyright (c) 1991 - 2011 BMC Software
 Inc. all rights reserved

 BMCVer=7.6.04
Appended Text=<null>
```

CONSOLE

```
List all connected users and last accessed time
  vkumar - Fri May 23 18:06:16 IST 2014
  ibrsduser - Fri May 23 19:01:20 IST 2014
  farooq.mohammed@in.unisys.com - Fri May 23 18:
50:11 IST 2014
  vidharva.nanda@unisys.com - Fri May 23 19:00:45
IST 2014
  mital.joshi - Fri May 23 19:01:25 IST 2014
  sriram.pattabhiraman@unisys.com - Fri May 23 19:
01:26 IST 2014
```

OUTPUT

```
Name of filter  HPD:ASI:Service_020_SetupFields01
Last modified by     vkk
owner is  dev_sleu
Action List is   5
Execution Order is  20
Permissions is  []
Else list is []
Primary form is      HPD:Incident Assoc Search
Properties are  []
```

Figure 5: Display of workflow information collected from the server

## VI. CONCLUSION

This tool will be useful in corporate environment for folks working with remedy log files for resolving the problems related to application with time constraint involved. The time spent on manually going through huge, confusing log files is saved. This time could be fruitfully spent on the solution of the problem rather than finding the cause of the problem. This indeed reduces the effort of the developers in solving the issues and increases the productivity in an organization.

## REFRENCES

[1] Dapeng Liu; Shaochun Xu; Huafu Liu “Using Log Files as Knowledge Bases “Industrial and Information Systems (ICIIS), in 6th IEEE International Conference onAdvanced Applied Informatics, 2011  On page(s): 130 – 135

[2] P.W.D.C. Jayathilake, “A Novel Mind Map Based Approach for Log Data Extraction” in 6th International Conference on Industrial and Information Systems,2011, ICIIS 2011, Aug. 16-19, 2011, Sri Lanka

[3] B. Beizer, Software Testing Techniques, 2nd Edition, 2 Sub. IntlThomson Computer Pr (T), 1990

[4] “Log4j.” [Online]. Available: http://logging.apache.org/log4j/1.2

[5] J. H. Andrews, “Testing using log file analysis: tools, methods, and issues,” in 13th IEEE International Conference on Automated SoftwareEngineering, 1998. Proceedings, 1998, pp. 157–166.

[6] J. H. Andrews, “Testing using log file analysis: tools, methods and issues,” Proc. 13th IEEE International Conference on AutomatedSoftware Engineering, Oct. 1998, pp. 157-166.

[7] James H. Andrews, “Testing using Log File Analysis:Tools, Methods, and Issues”

[8] J.H.Andrews. Theory and practice of log file analysis. Technical Report 524, Department of Computer Science,Universityof Western Ontario, May 1998

[9] BMC Remedy Action Request System 7.6.04 Concepts Guide

[10] L.K. Dillon, G. Kutty, L. E.Moser, P.M.Melliar-Smith, and Y. S. Ramakrishna. Graphical interval logic for specifying concurrent systems. ACM Transactions on Software Engineeringand Methodology, 3(2):131–165, April 1994.