



Erlang Training and Consulting Ltd

Exago

Aniko Nagyne Vig and Atilla Erdodi

Stockholm, 13th Nov 2009



What can we use log files for?

The ideal audit log files capture the essence of the system. Contains a permanent record of the most important key points of the life of the working system.

There are many other tool around to analyse them: mainly to provide statistics, make it more "human readable"



Why Exago is different?

In Exago we have a different goal:

Use the audit logs to test your live system and your logs.

Even after a tested system is deployed and it is in production for years, from time to time there is a need to prove the system behaves correctly in specific cases, or an error needs to be located. The first attempt is to use the log files. We would like to ease this process with Exago.

Copyright 2008

The Erlang logo, featuring the word "Erlang" in a stylized, cursive red font.

A solution

Exago: An audit log monitoring tool.

Prototype by Hans Svensson, Chalmers Univ.

Development by Erlang Training and Consulting Ltd.

Copyright 2008

The Erlang logo, featuring the word "Erlang" in a stylized, cursive red font.

How to describe the expected behaviour?

- Using Prolog or some proprietary language is a solution, but needs time to learn ...
- Check it against a state machine!
- Simple yet powerful.
- By adjusting the granularity, we can check more complex properties as well ...

Copyright 2008



Idea

- Reconstruct what happened and create a series abstract commands. During this process the dependencies and connections between the fields of log files can be checked.
- See if the system behaved properly, using some sort of model checking - building an abstract state machine for the system. Every log entry will represent an edge in the state machine.
- Keep the tool general enough to allow configuring different level of detail checks, creating different state machines from the state system logs.

Copyright 2008



How can we transform the logs to a state machine?

Lines in the log files → ... → State machine

- Based on the content of one or more entries will be one edge in the state machine
- Entries in the log files needs to be grouped to form one execution instance of the model

Copyright 2008



Terminology

- **Transaction:**
 - A set of events that describe a state change
 - To be abstracted to a single command
- **Session:**
 - A series of transactions to be checked against a state machine

NOTE:

- These terms depend on the level of detail,
- not necessary mean the same from the system's and tool's perspective.

Copyright 2008



Available interfaces:

- Erlang record,
- command line,
- GUI built on wxWidgets,
- (planned Emacs and Eclipse integration).

Copyright 2008

The Erlang logo, featuring the word "Erlang" in a stylized, red, cursive font.

Results so far

- Two industrial case studies, from a different perspective
- Production system for years: *(TPSG)*
 - Took approximately 2 day to create the model
 - After spent several years on testing the system, we still managed to found bugs just by analysing log files!
- Using Exago as a tool of development *(SMSC)*
 - In progress ...
 - Helping quality assurance

Copyright 2008

The Erlang logo, featuring the word "Erlang" in a stylized, red, cursive font.

To use Exago, you have to:

- Get some log files from the production system,
- Specify relations between the files and their formats,
- Provide abstraction and validation functions for the processing,
- Present an abstract model of the system as a state machine.

Copyright 2008



Log files

Preconditions:

- must be CSV formatted (plans for supporting other formats as well),
- must provide „enough information”.

Exago specific:

- we specify a set of log files at a time using wildcards

Copyright 2008



Log files

Audit files | Transaction abstrfun | Transaction validfun | Session abstrfun | State machine

Files

- esas_mt_sms*.audit
- esas_mtcq_sms*.audit
- esas_delrep*.audit
- esas_uds_recredits*.audit
- failed_mt_sms*.audit

Add Remove

File details

Directory:

Filename prefix: esas_delrep

Filename suffix: .audit

Transaction id:

Session id: 3,5

Timestamp: 1

Abstract value: 14

Foreign keys:

Id	Aggr	FKey
message_re session		2,5,9,14,11

Copyright 2008

Erlang

Exago - configuration

Giving the details of individual log files:

- Directory inside the given base one
- Filename prefix and suffix
- Transaction and session identifiers
- Abstract value
- Foreign keys

Copyright 2008

Erlang

Important Values for the analysis

- Timestamp
- Session id
- Transaction id
- Abstract value

Specify the values for the tool:

- Describe them using a list of integers denoting the positions in the CSV log files
- eg.: [3,11,5]

Copyright 2008

Erlang

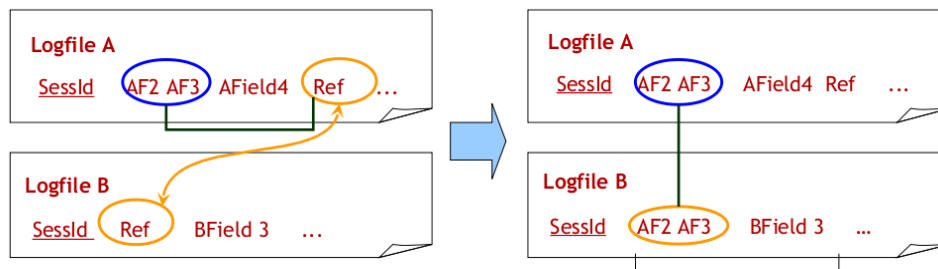
Relations

Configuration for logfile A:

```
{ ...  
  session_id = [1],  
  fkeys = [{ref_id, [2,3], [5]}]  
  ... }
```

Configuration for logfile B:

```
{ ...  
  session_id = [1],  
  abstract_value = [{ref_id, [2]}, 3]  
  ... }
```



Copyright 2008

Erlang

Relations

The screenshot shows a software interface with several tabs: "Audit files", "Transaction abstrfun", "Transaction validfun", "Session abstrfun", and "State machine". The "Audit files" tab is active, displaying a list of files on the left and "File details" on the right. The file list includes "esas_mt_sms*.audit", "esas_mtcq_sms*.audit", "esas_delrep*.audit", "esas_uds_recredits*.audit", and "failed_mt_sms*.audit". The "File details" section contains fields for Directory, Filename prefix (esas_delrep), Filename suffix (.audit), Transaction id, Session id (3,5), Timestamp (1), and Abstract value (14). Below these fields is a "Foreign keys" table with columns "Id", "Aggr", and "FKey". The table contains one entry: "message_re session" with "2,5,9,14,11" in the "FKey" column. This entry is circled in red.

Id	Aggr	FKey
message_re session		2,5,9,14,11

Copyright 2008

Erlang

Abstraction

Parsed and resolved log entries

→ *aggregate and abstract* →

Transactions

→ *filter* →

Filtered transactions

→ *abstract and group* →

Sessions

NOTE: Transaction abstraction and validation are optional

Copyright 2008

Erlang

State machine

The screenshot shows a software interface for configuring a state machine. It has a menu bar with 'Audit files', 'Transaction abstrfun', 'Transaction validfun', 'Session abstrfun', and 'State machine'. The 'State machine' menu is active, showing three sub-sections: 'States', 'Transitions', and 'Options'.

State nr	State name
0	Init
1	State 1
2	State 2
3	State 3
4	State 4
5	State 5
6	State 6

From nr	To nr	Command	Timeout
0	6	mt_sms_failed	
6	5	mt_sms_del_failed	
0	1	mt_sms_accepted	
1	5	mt_sms_del_succ	
1	5	mt_sms_del_failed	
0	2	mtcq_sms_billed	
2	3	mt_sms_accepted	
3	5	mt_sms_del_succ	
3	4	mt_sms_del_failed	
4	5	mtcq_sms_recredit	

Options

Terminal states:

Good terminal states:

Initial state:

Copyright 2008

Output

- Generated HTML output,
- Errors and warnings are listed,
- Failed sessions are listed, failed command highlighted,
- Generated state machine with graphviz, with highlighted terminal state.

Copyright 2008

What happens after the analysing the results?

- In optimal case the result show up no problems, so the system and logging is verified.
- If any problem was found, the reason needs to be find, it was in the logging procedure, or in the functional part of the system.
 - Further knowledge can be involved by not just using the tools, but support engineers or developers
 - Further detailed debugging using on-line tracing on the specific problem on the live system by onviso for example
- The found and fixed issue can result in a new unit or system test case, and can be used to test with continuous integration if the tool was used during the development phase

Copyright 2008



Pros and cons

Main advantages:

- Don't need access to the live system
- Can analyse errors without reproducing it, just check the log entries at the time the failure occurred

Disadvantages:

- Limited usability: logging must meet certain requirements
- The information is limited in the audit log files

Copyright 2008

