Erlang Solutions Ltd.

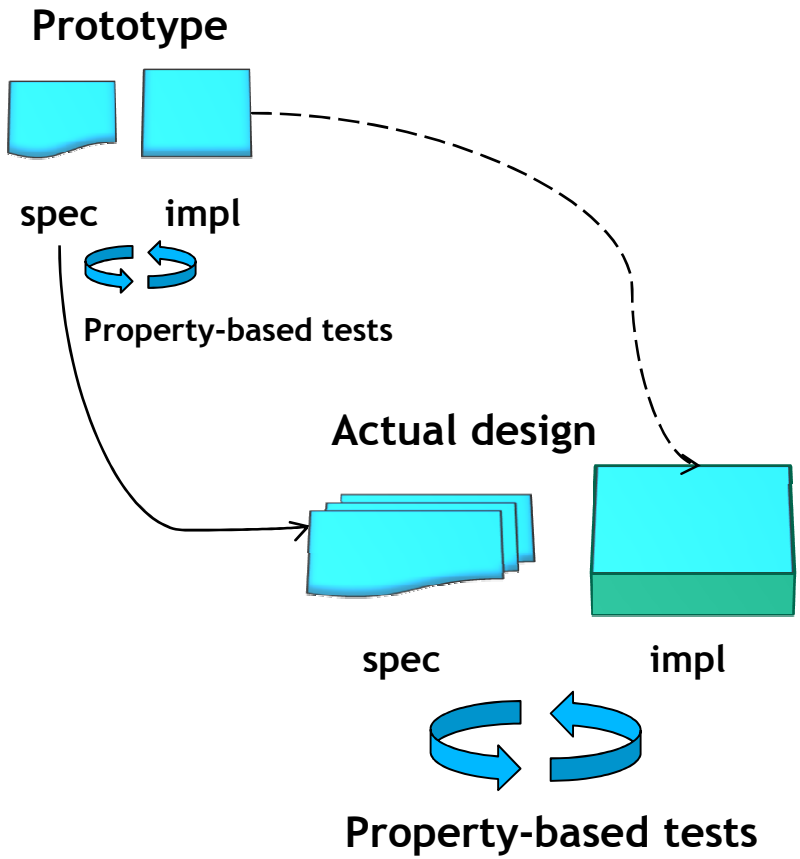
Onviso and Exago

Tracing and log analysis in multiple-node environments

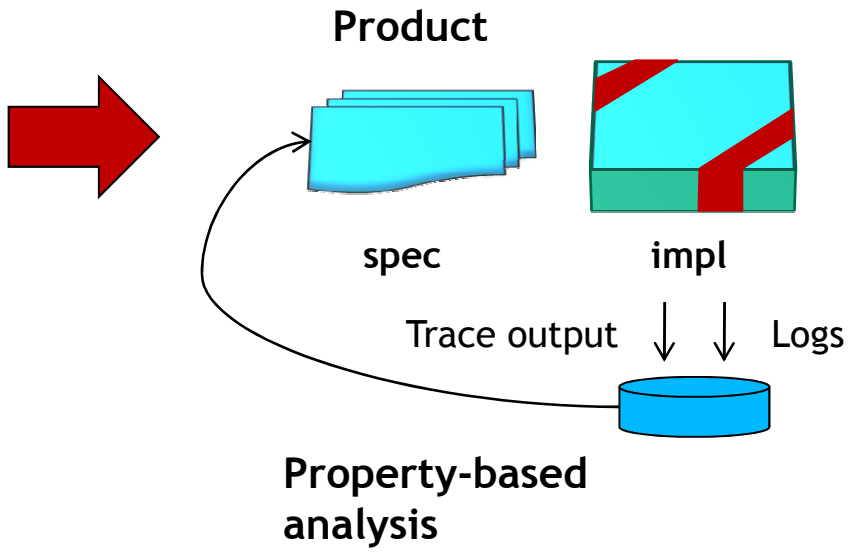
Ulf Wiger, Bartłomiej Puzoń, Atilla Erdődi
Erlang Solutions Ltd

Erlang Factory, London, 10 June 2010

10,000 ft Perspective



- Abstract properties are...
  - More portable
  - More stable
  - More versatile



ProTest property based testing

Erlang

The tracing and log analysis problem

The Protest project:
EU-funded research on Property-based Testing

Tracing and log analysis work package:

- How to conduct safe and efficient run-time trace analysis on distributed systems?
- How to do advanced post-mortem log analysis?
  - (or indeed log analysis on running systems?)
- Eventually reuse high-level properties from testing

ProTest property based testing

Erlang

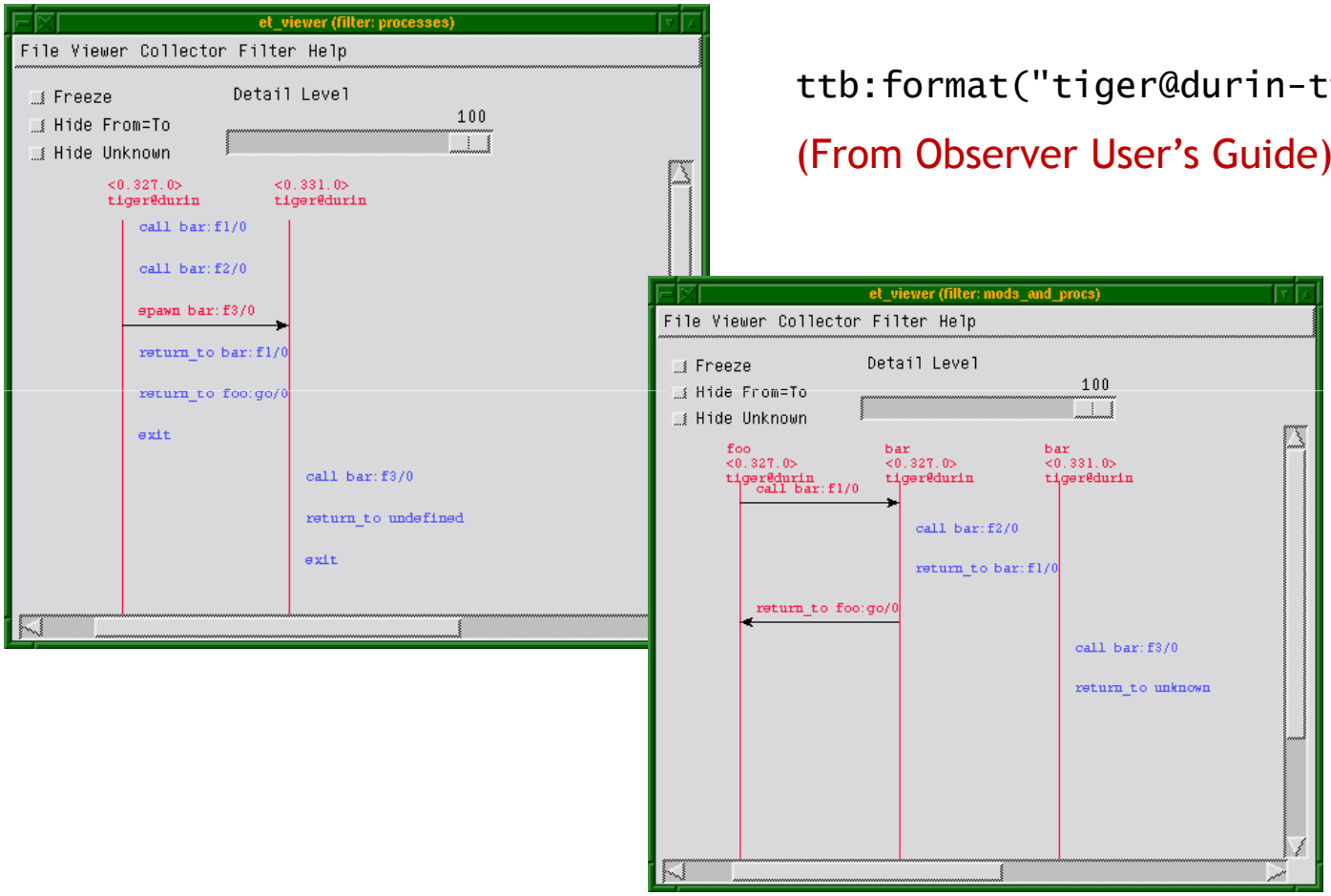
Tracing support in Erlang

- The trace() BIFs
  - Low-level trace message generation
  - Dynamic control using Match Specifications
- The DBG application
  - Command-line wrappers around the trace BIFs
  - (Redbug, a dbg alternative made by Mats Cronqvist)
- Observer, Trace Tool Builder, etop, et, pman
  - Various loosely connected utilities
- Percept, eprof, cprof, fprof, instrument
  - Profiling tools with different characteristics

ProTest property based testing

Erlang

Lots of functionality, Hard to Grep



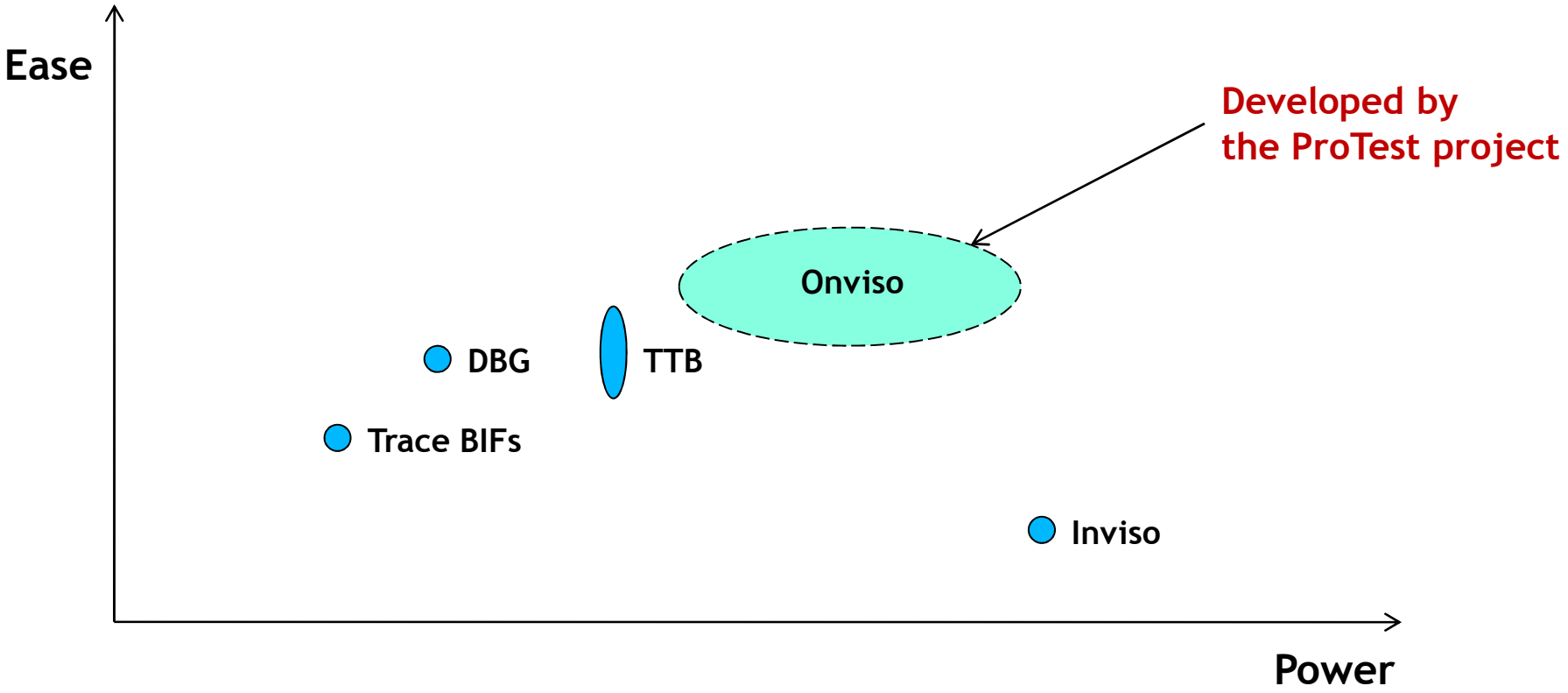

```
ttb:format("tiger@durin-ttb", [{handler, et}])
```

(From Observer User's Guide)

ProTest property based testing

Erlang

Searching for the Sweet Spot



ProTest
property based testing

Erlang

Multi-node Tracing in OTP

- dbg
  - `dbg:n(Nodename)` includes a node in the traced set
- ttb
  - `ttb:tracer(Nodes, Options)` sets up a multi-node trace
  - `ttb:stop([fetch])` fetches logs from traced nodes
  - `ttb:format(FileOrDir [,Options])` merges/processes the trace logs
  - Meta-tracing, save config, run config, sequence trace support
- inviso
  - Adds overload protection, heterogeneous tracing, return value matching, autostart, trace cases, ...
  - Steep learning curve

ProTest property based testing

Erlang

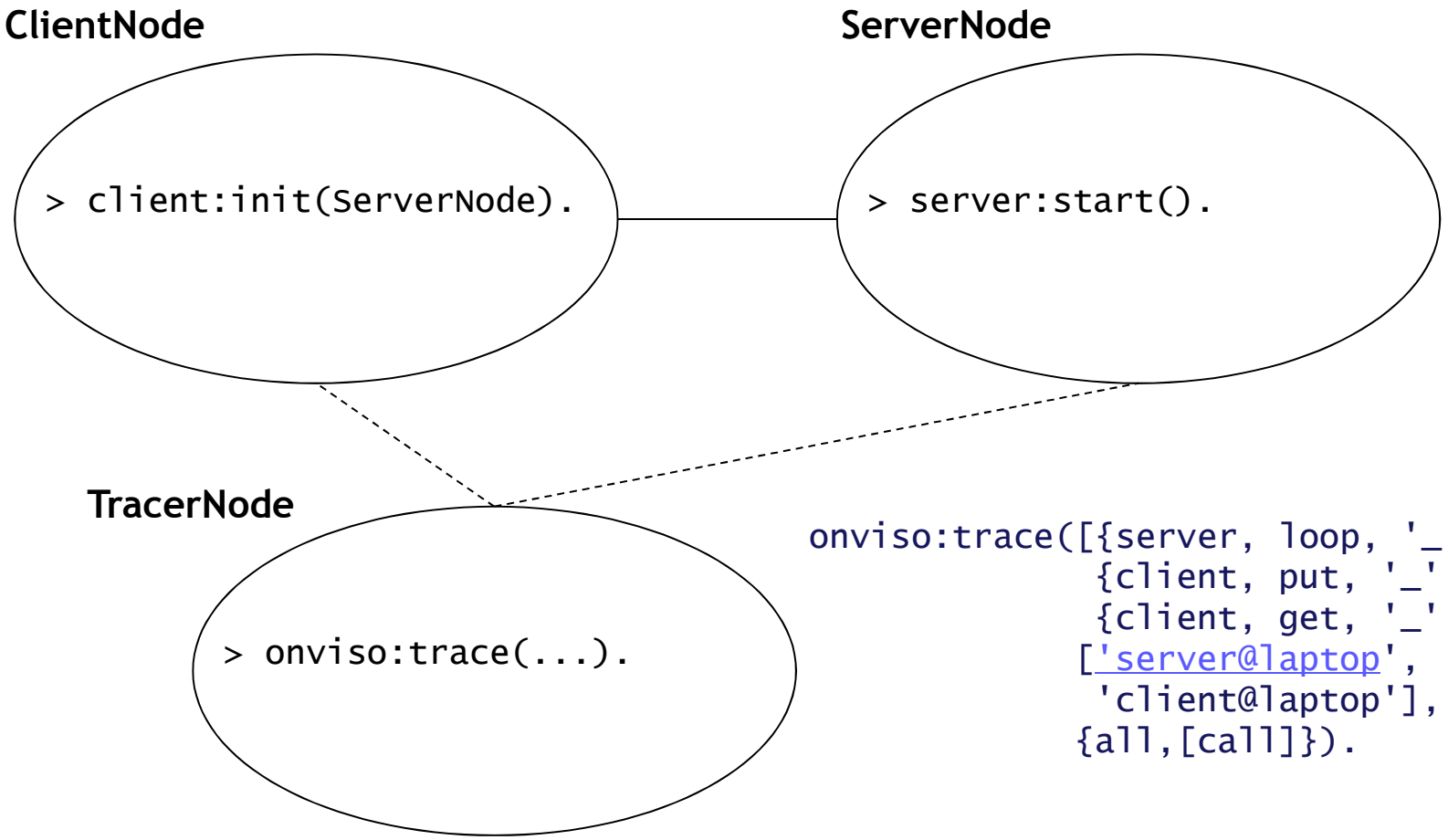
Onviso *(means absolutely nothing)*

- User-friendly API to Inviso *(latin: "I inspect")*
  - Set up and run tracing using only two commands
  - Shortcuts for commonly used trace patterns (inspired by Redbug)
- Additional functionality
  - Non-destructive merge of trace logs
  - Useful defaults for merging and overload protection
  - Trace node automatically reconnects to restarting target nodes
  - "cli", a wizard-like aide to defining trace cases
- Status: Work in Progress
  - http://github.com/esl/onviso-dev

ProTest property based testing

Erlang

Demo – Starting the Nodes




```
onviso:trace([{server, loop, '_', []},
              {client, put, '_', []},
              {client, get, '_', return}],
             ['server@laptop',
              'client@laptop'],
             {all,[call]}).
```


ProTest property based testing

Erlang

Interrupting a Trace

- One of the nodes can be restarted:

```
client@laptop> init:restart().
```

```
client@laptop> client:init('server@laptop').
```

- By default Onviso will reconnect and resume tracing on the client node.
- If the node restarts abruptly, some of the trace data may be lost (as the trace buffers might not be flushed to the files).
  - Inviso (and thus, Onviso) can handle incomplete trace logs.

Stopping a Trace

- Every trace call returns a trace reference identifier. This id can be used to stop or merge a trace

```
> onviso:stop(Id).
```

- The traces are collected to files and distributed back to the Inviso control node

Onviso Command line interface

- Example of a higher-level trace tool
- Help testers and support staff define and/or execute trace cases

```
(inviso@debian)6> cli:start().
Onviso Demo GUI
=======================================


 > Main Menu
---------------------------------------
 1) Add trace case
 2) List/Run trace cases
 3) Save configuration to file
 4) Load configuration from file
 5) Set the magic cookie
 6) Exit
[Q] Choice [1-6] : 6

 Exiting...ok
```

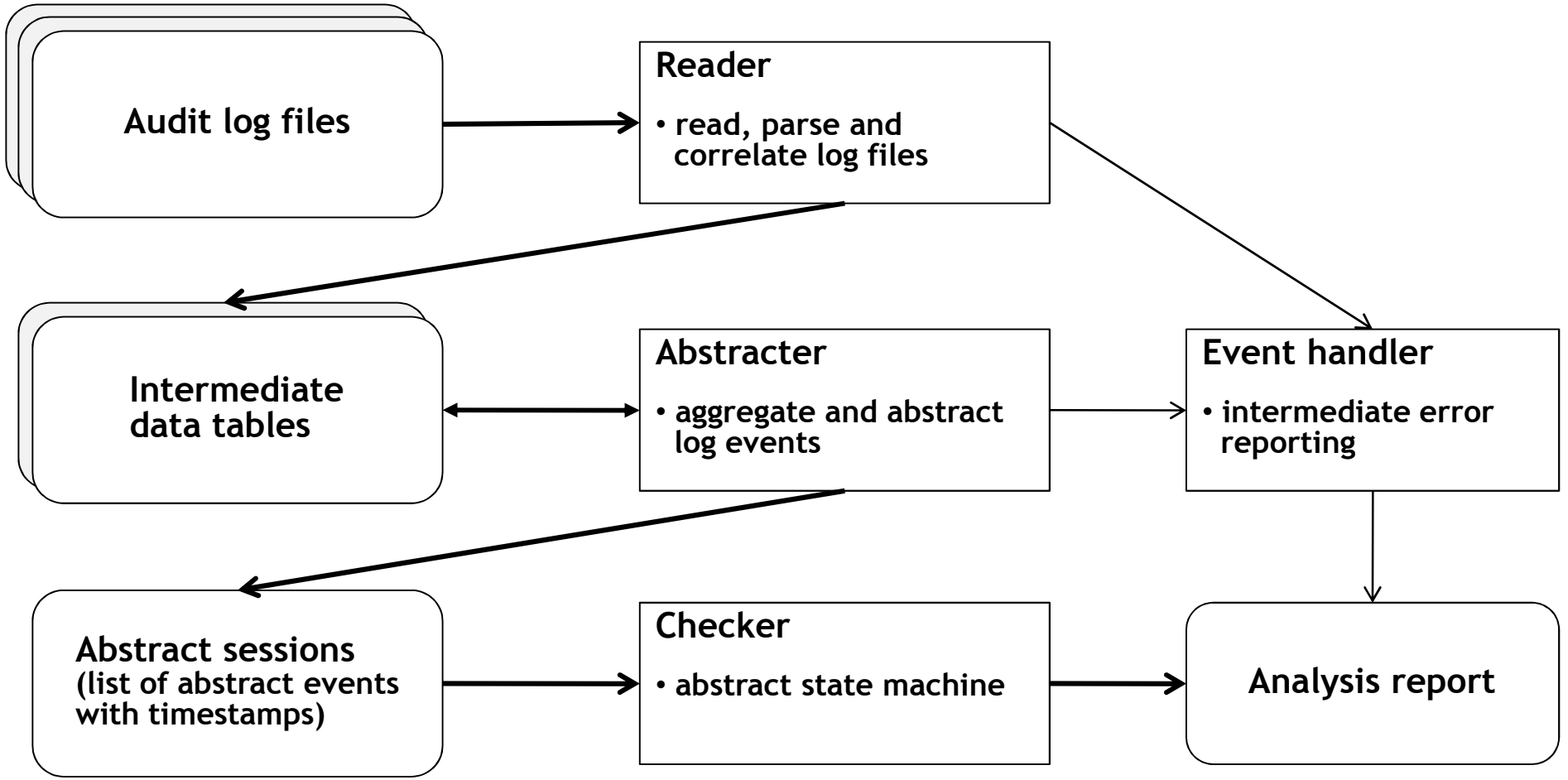

ProTest property based testing

Erlang

εξαγω – Ancient Greek: “bring forth”

EXAGO

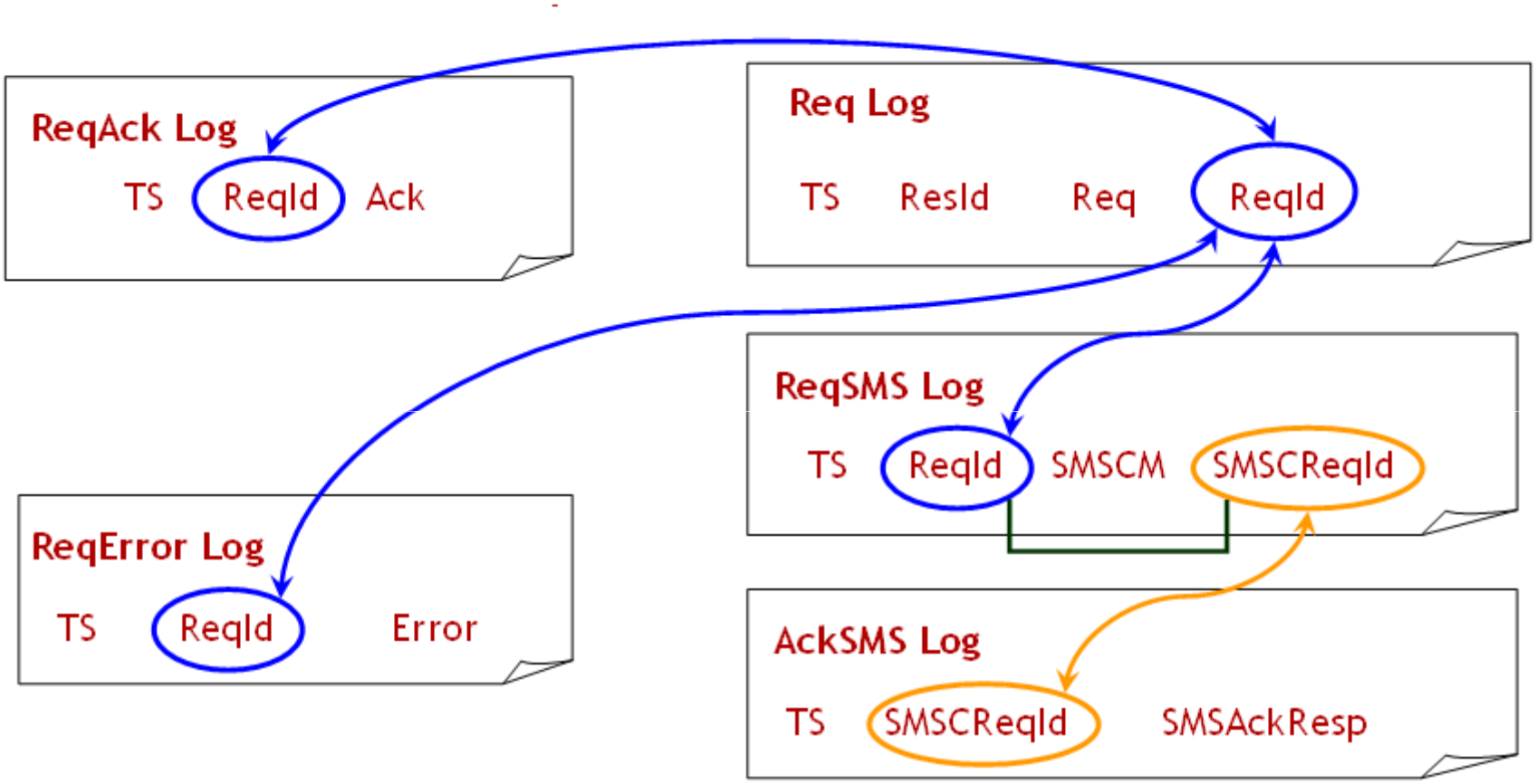
A “log mining” Approach



ProTest property based testing

Erlang

Log Correlation Example



ProTest property based testing

Erlang

Exago Status

- http://github.com/esl/Exago
- Two case studies
  - Finding bugs in a well-tested stable system
  - Using Exago in the early stages of development
- Need more case studies
- Work on scalability
- Investigate applying QuickCheck's Temporal Relations

ProTest property based testing

Erlang

Case Study: SMS Gateway

```
[{"2008-08-07_05:34:10:862",mtcq_sms_billed},
 {"2008-08-07_05:34:15:864",timeout},
 {"2008-08-07_05:34:15:864",{mt_sms_del_failed,{"timedout"}}},
 {"2008-08-07_05:34:21:275",mt_sms_accepted},
 {"2008-08-07_05:34:29:010",mt_sms_del_succ}]
```

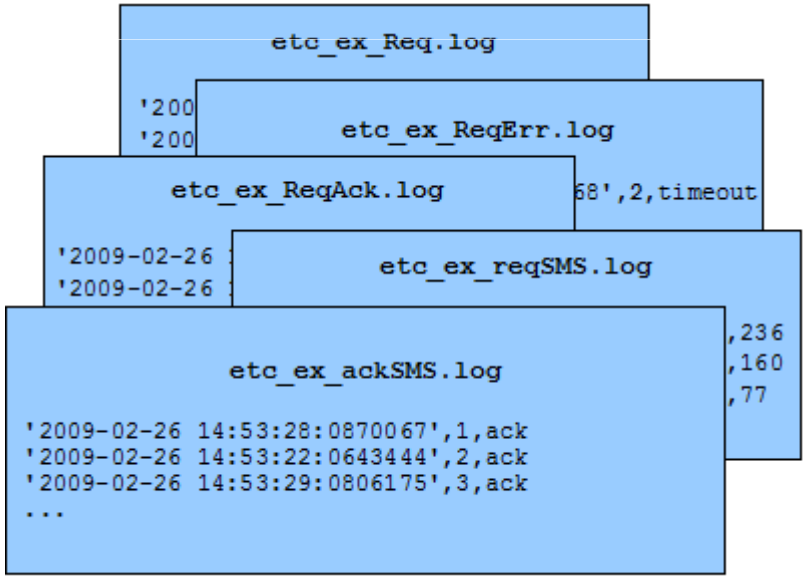
- Gateway times out, delivers a failure report to user
- SMSC finally reports successful delivery, gateway forwards it
- User gets conflicting reports + could interfere with SMS retry
- 2 occurrences among 20,000 sessions in the log
  - Exago pilot duration: 2 days
  - System had been in production for two years...

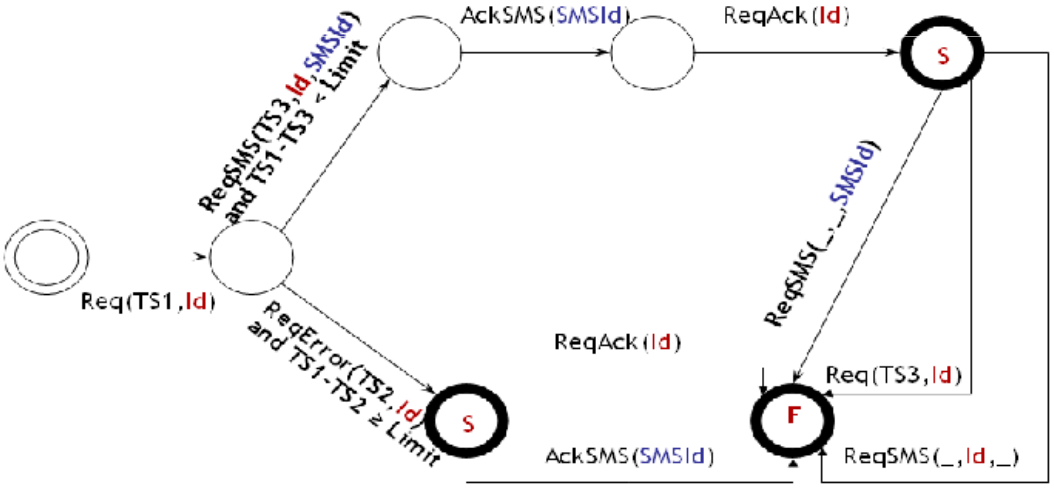
ProTest property based testing

Erlang

Example

- A simplified, ideal SMS Gateway System
  - http://github.com/esl/Exago/blob/master/apps/exago/test/etc_example_gen.erl
- CSV log files are artificially generated





ProTest property based testing

Erlang

Parse and Resolve

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang

Parse and Resolve

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

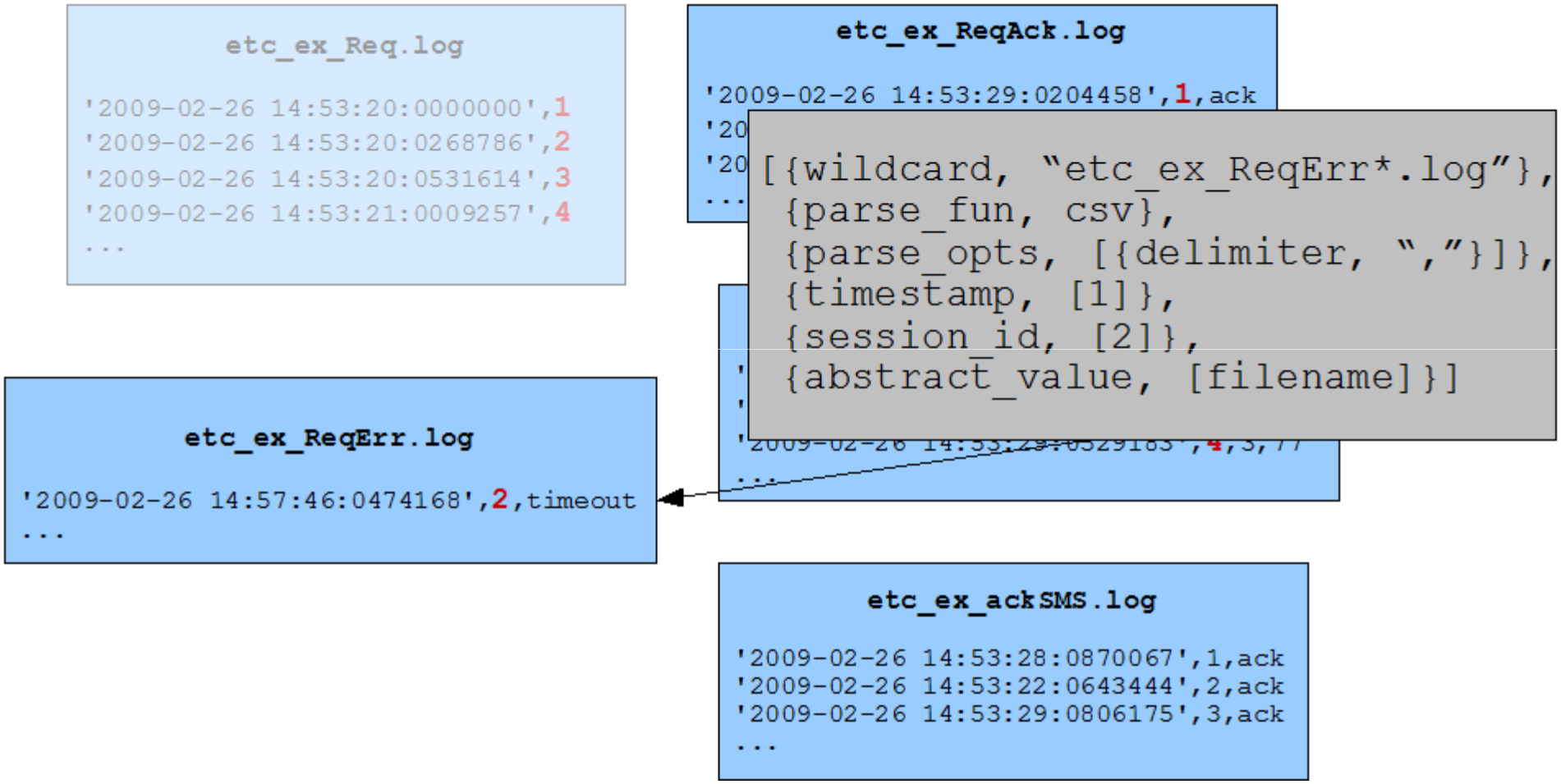
etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang

Parse and Resolve



ProTest property based testing

Erlang

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```

ProTest property based testing

Erlang

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

Defines a mapping between the ids!

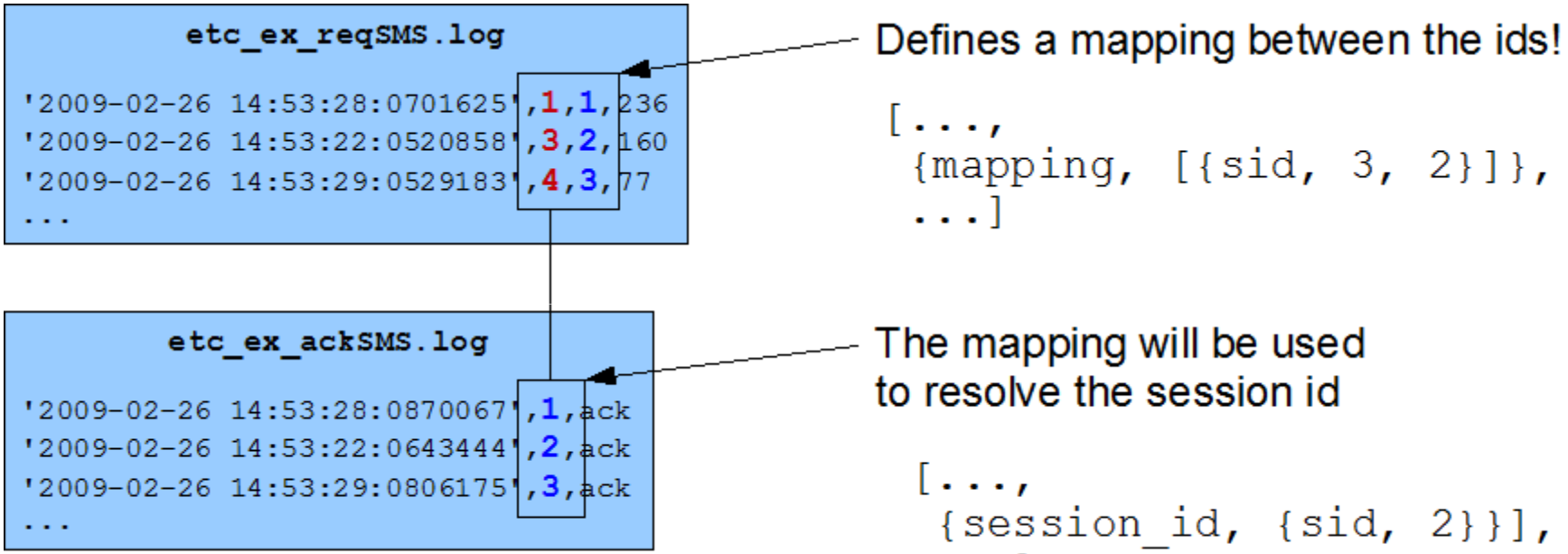
```
[...,
 {mapping, [{sid, 3, 2}]},
 ...]
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang



ProTest property based testing

Erlang

Aggregate

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang

Aggregate

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang

Aggregate

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang

Aggregate

etc_ex_Req.log

```
'2009-02-26 14:53:20:0000000',1
'2009-02-26 14:53:20:0268786',2
'2009-02-26 14:53:20:0531614',3
'2009-02-26 14:53:21:0009257',4
...
```

etc_ex_ReqAck.log

```
'2009-02-26 14:53:29:0204458',1,ack
'2009-02-26 14:53:22:0751754',3,ack
'2009-02-26 14:53:29:0823714',4,ack
...
```

etc_ex_ReqErr.log

```
'2009-02-26 14:57:46:0474168',2,timeout
...
```

etc_ex_reqSMS.log

```
'2009-02-26 14:53:28:0701625',1,1,236
'2009-02-26 14:53:22:0520858',3,2,160
'2009-02-26 14:53:29:0529183',4,3,77
...
```

etc_ex_ackSMS.log

```
'2009-02-26 14:53:28:0870067',1,ack
'2009-02-26 14:53:22:0643444',2,ack
'2009-02-26 14:53:29:0806175',3,ack
...
```


ProTest property based testing

Erlang

Abstract

- Transaction abstraction (optional)
  - Can group related events into a single event

```
[...,
 {trans_abstr, fun trans_abstr/1},
 ...]
```

- Session abstraction
  - Convert actual log events to symbolic values

```
[...,
 {sess_abstr,
  fun(Trs) -> lists:map(fun abstr_sms/1, Trs) end},
 ...]
```

```
trans_abstr(EventList) ->
    case EventList of
        [{TsReq, login_req, UserId},
         {_TsAck, login_ack, UserId}] ->
            {TsReq, login_succ, UserId};
        ...
    end
```

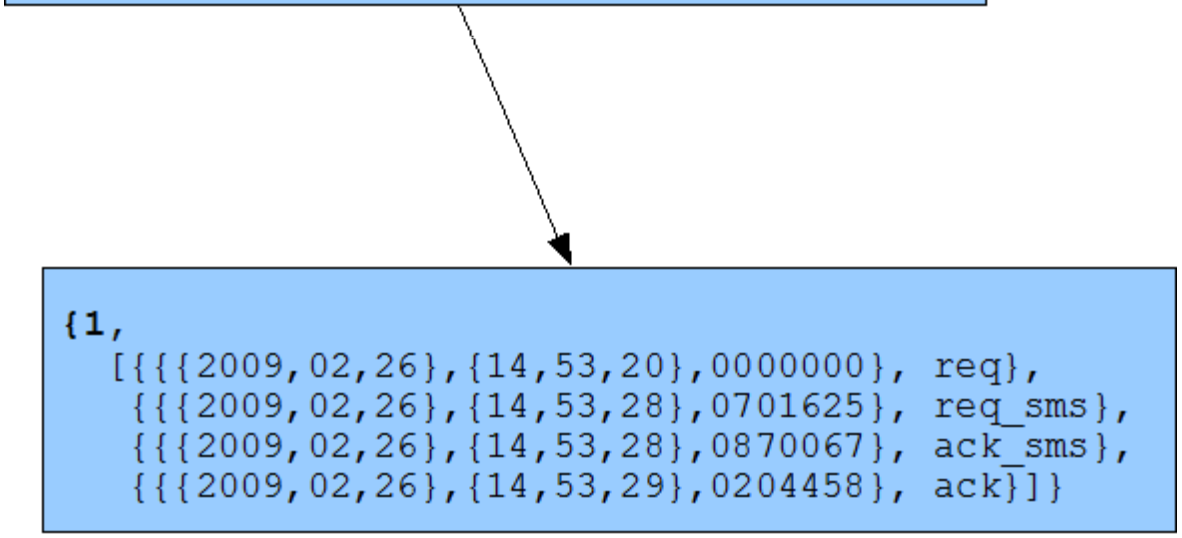
```
abstr_sms({Ts,{File}}) ->
  {match,[Type]} =
    re:run(File,"([^_]+)\\.log",
           [{capture,[1],list}]).
    {Ts, proplists:get_value(
              Type, [{“Req”, reg}.
                     {“ReqSMS”, req_sms},
                     {"AckSMS“, ack_sms},
                     {"ReqAck“, req_ack},
                     {"ReqErr“, req_err}]}.
```


ProTest property based testing

Erlang

Result of aggregation and abstraction

```
etc_ex_Req.log
   '2009-02-26 14:53:20:0000000',1
etc_ex_ReqAck.log
   '2009-02-26 14:53:29:0204458',1,ack
etc_ex_reqSMS.log
   '2009-02-26 14:53:28:0701625',1,1,236
etc_ex_ackSMS.log
   '2009-02-26 14:53:28:0870067',1,ack
```



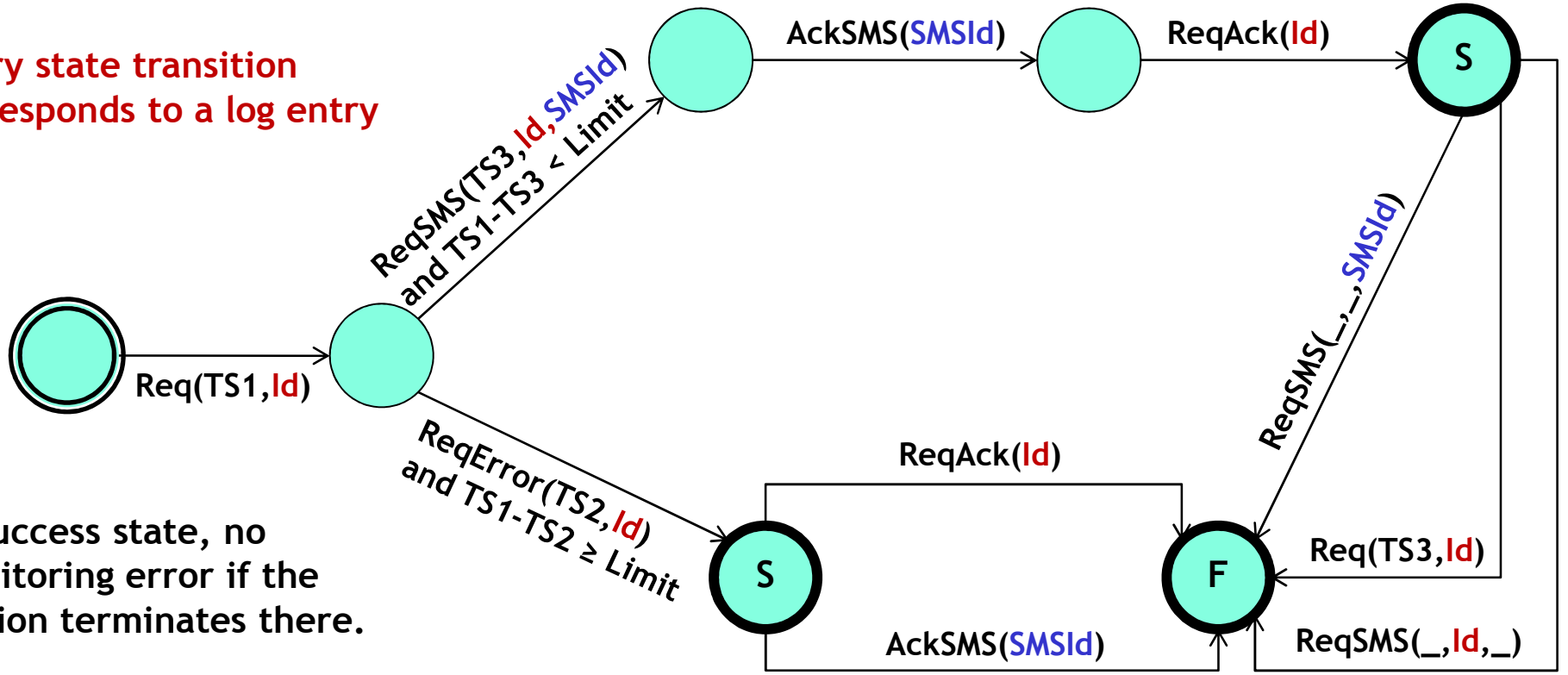
```
{1,
   [{{{2009,02,26},{14,53,20},0000000}, req},
    {{{2009,02,26},{14,53,28},0701625}, req_sms},
    {{{2009,02,26},{14,53,28},0870067}, ack_sms},
    {{{2009,02,26},{14,53,29},0204458}, ack}]}
```


ProTest property based testing

Erlang

Check

Every state transition corresponds to a log entry



S: Success state, no monitoring error if the session terminates there.

F: Bad state, monitoring error if the session terminates there.

ProTest property based testing

Erlang

Check

```
{statem,
  [{states, [0,1,2,3,4,5,6]},
   {trans,  [{0,1,req},
             {1,2,req_sms, {lt,30}},
             {2,3,ack_sms},
             {3,4,req_ack},
             {1,5,req_err, {geq,30}},
             {5,6,req_ack},
             {5,6,ack_sms},
             {4,6,req},
             {4,6,req_sms}]},
   {terminal,[4,5,6]},
   {good,    [4,5]}]
}.
```

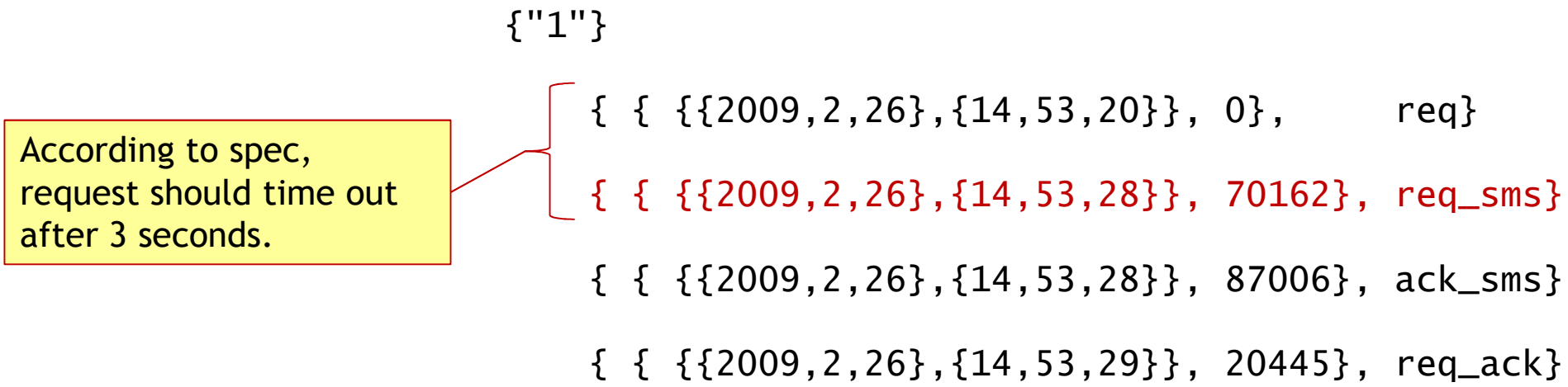
- State machine specified as a Labelled Transition System
- Time constraints for transitions in 1/10th sec
- Not all terminal states are “good” states

ProTest property based testing

Erlang

Check

- No matching transition within time constraint for req_sms in state 1



ProTest property based testing

Erlang

Summary

- A language-agnostic, high-level, multi-log analysis tool
- Pluggable with custom parsers, filters and checkers
- Has found bugs in mature commercial systems with little effort
- http://github.com/esl/Exago

- Future work:
  - Scalability
  - Integration with QuickCheck
  - Test on more products from different domains

ProTest property based testing

Erlang

Thank you

Questions?