



Easy to use, no restrictions

RefactorIt

terminal build interactive
 possibilities Emacs Linux shell OS X
 source scripts graphical desktop-applications
 Emacs Linux shell OS X
 terminal build interactive
 possibilities Emacs Linux shell OS X
 source scripts graphical desktop-applications

Sharing information between the team members

Who are we?

Several of us (including myself) have been using RefactorIt for some time now and we are happy to share our experience with you. We are currently working on the next version of RefactorIt and we are looking for feedback from our users. If you are interested in using RefactorIt, please contact us at refactorit@refactorit.com. We are currently working on the next version of RefactorIt and we are looking for feedback from our users. If you are interested in using RefactorIt, please contact us at refactorit@refactorit.com.

What is RefactorIt?

RefactorIt is a tool that helps you to refactor your code. It is designed to be easy to use and to work with a wide range of programming languages. It is currently in beta and we are looking for feedback from our users. If you are interested in using RefactorIt, please contact us at refactorit@refactorit.com.

Features

- Supports a wide range of programming languages
- Easy to use and intuitive interface
- Supports a wide range of programming languages
- Easy to use and intuitive interface

User interfaces

How to setup RefactorIt

RefactorIt is available for Windows, Linux, and Mac OS X. It is easy to install and use. For more information, please visit our website at refactorit.com.

Some advantages

- Easy to use and intuitive interface
- Supports a wide range of programming languages
- Easy to use and intuitive interface
- Supports a wide range of programming languages

Industrial applications

RefactorIt is used by several large companies, including Microsoft, IBM, and Oracle. It is a powerful tool for refactoring code in a wide range of programming languages.

Experience

RefactorIt is a powerful tool for refactoring code. It is easy to use and intuitive. For more information, please visit our website at refactorit.com.

refactorit.com

News

Pattern Discovery
<http://code.google.com/p/refactorit/>

Demo time

<http://plm.inf.ed.ac.uk/refactorit/>

Single analysis view features

- Supports a wide range of programming languages
- Easy to use and intuitive interface
- Supports a wide range of programming languages
- Easy to use and intuitive interface

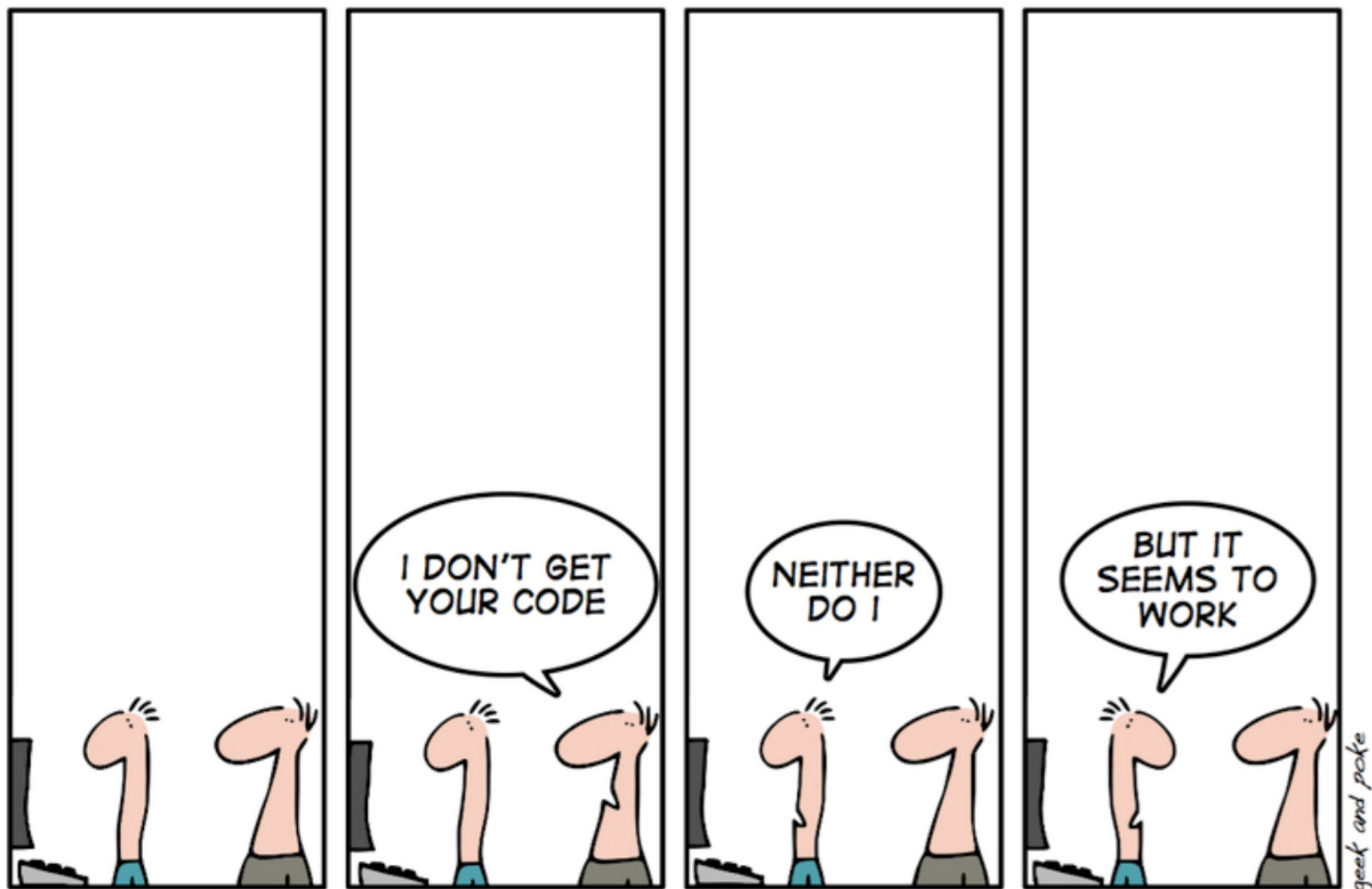
Advanced Features

- Semantic Queries & Parametrised Queries
- Collaborative work support
 - query & graph sharing
 - stateful links
- Investigations

And a bit more advanced stuff...

- Centralised management support
 - restricted mode for users
 - web or console based administration

Q & A



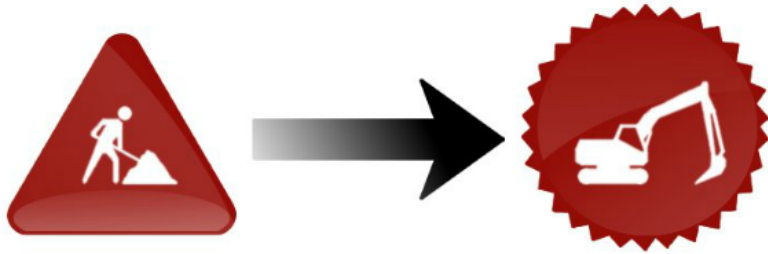
THE ART OF PROGRAMING

We do!

RefactorErl

Effective software maintenance

grokking



investigations

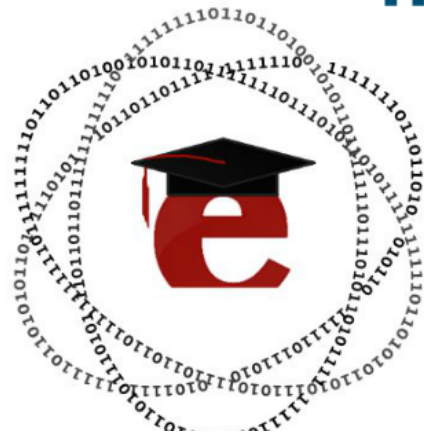
code duplicates

dependencies

metrics

refactoring

clustering



Knowledge sharing



Who are we?



Started as a refactoring project

University Staff &

Supports code comprehension

PhD, MSc, BSc students

ELTE-Soft R&D staff

Open Source! Try it!

refactorerl.com

Ericsson-ELTE Software Technology Lab (2011)

Contact:

tothmelinda@elte.hu

Open Source! Try it!

refactorerl.com

Ericsson-EL

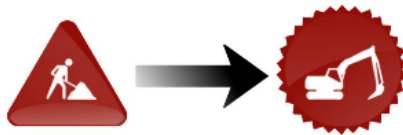
Contact:

tothmelinda@elte.hu

What is RefactorErl?

Understand legacy code

Effective software maintenance



Knowledge sharing

Refactoring

Check code complexity/quality

in Erlang
for Erlang



Features

Compile-time analysis of

Functions, variables, records, etc.

Lifetime, scope, visibility

Static and dynamic references

Side-effects

Data-flow, control-flow

Dynamic function calls

Hidden dependencies

Program comprehension

Semantic queries

Software complexity metrics

Bad smell detection

Duplicated code detection

Clustering

Dependency visualisation

... and more than 20 refactoring transformations.

Program comprehension

Semantic queries

Software complexity metrics

Bad smell detection

Duplicated code detection

Clustering

Dependency visualisation

Easy to use, no restrictions

A word cloud centered around the text 'RefactorErl'. The words are in various sizes and orientations, primarily in shades of brown and orange. The largest words are 'RefactorErl', 'Linux', 'Emacs', 'shell', and 'OSX'. Other visible words include 'terminal', 'build', 'interface', 'easily', 'support', 'offers', 'IDE', 'run', 'interactive', 'graphical', 'desktop application', 'plugins', 'open', 'Windows', 'web-based', 'possibilities', 'Solaris', 'source', and 'scriptable'.

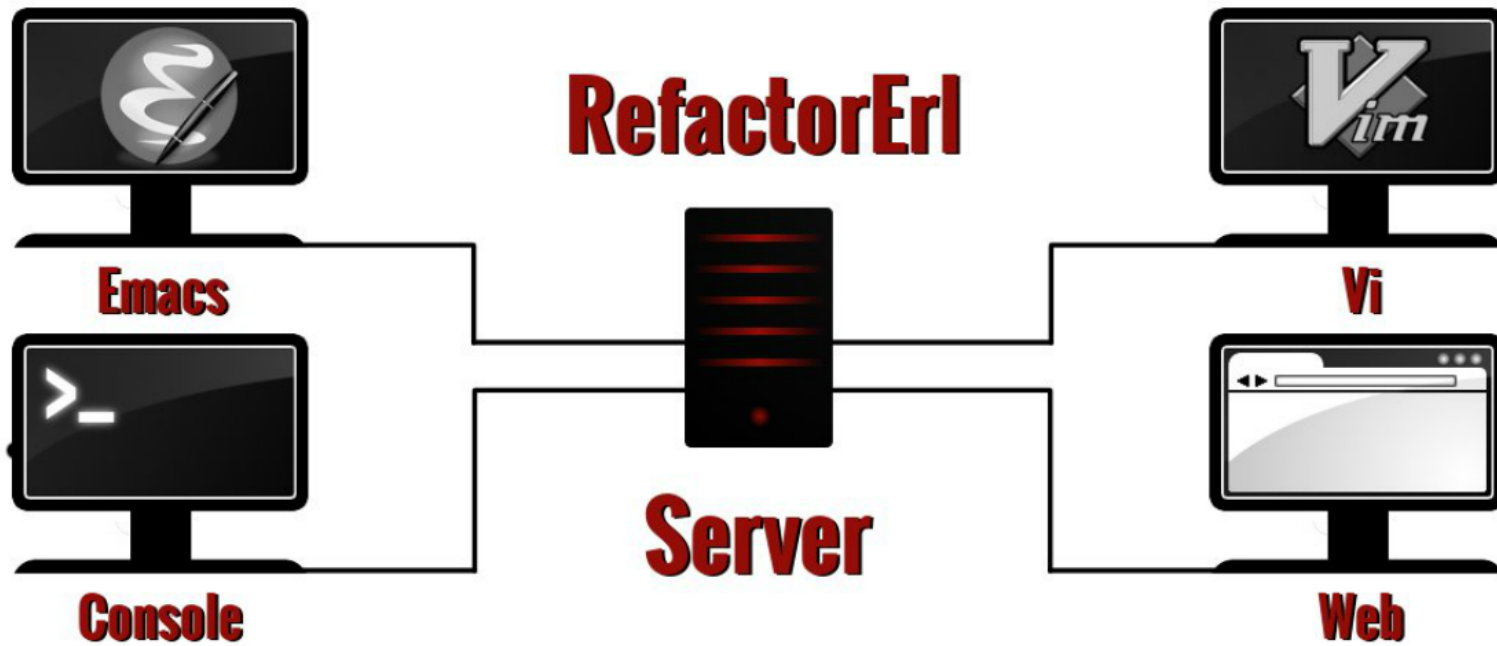
RefactorErl

terminal build interface easily support offers IDE run interactive graphical desktop application plugins open OSX

possibilities Solaris source

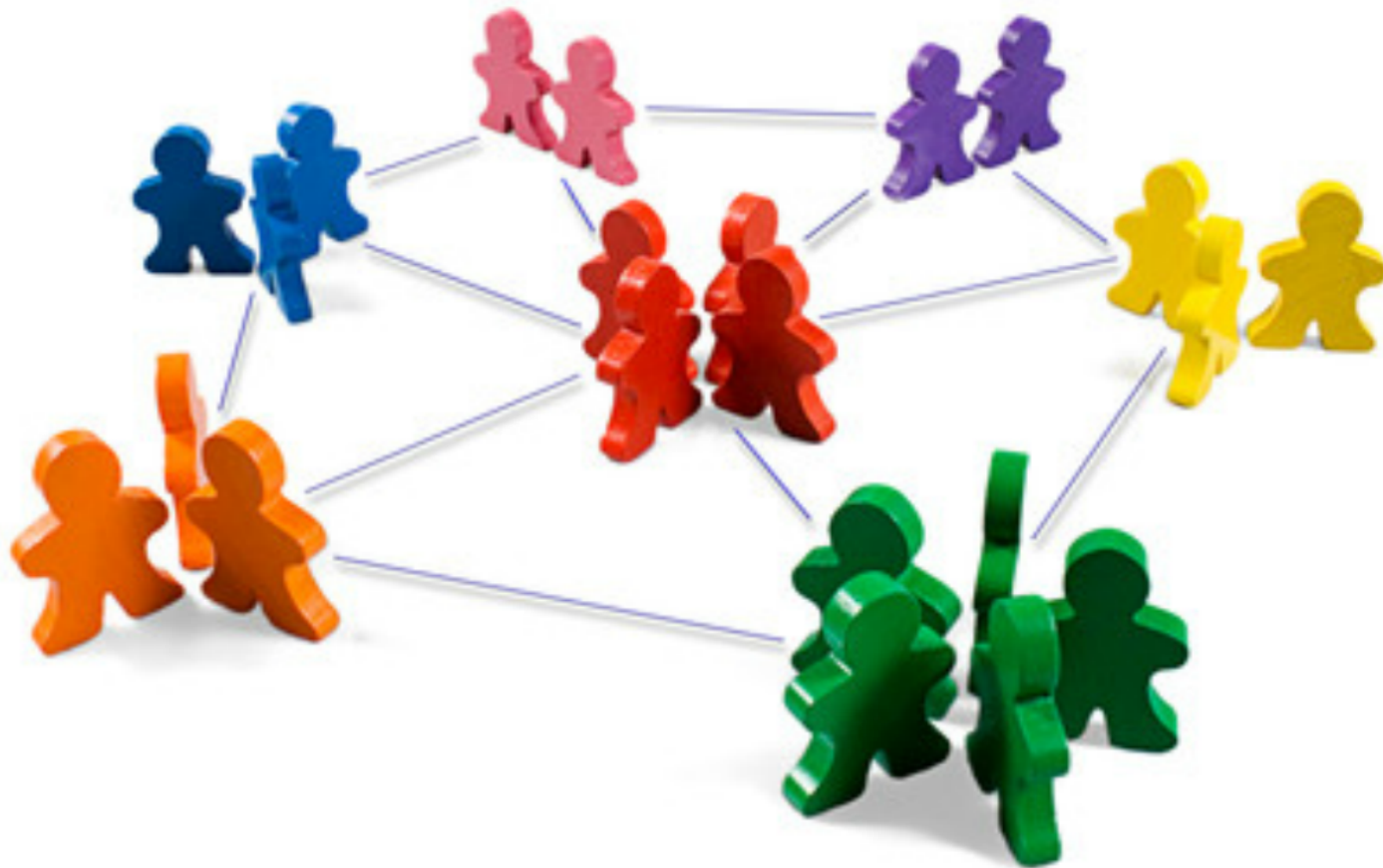
Windows web-based Emacs Linux scriptable

User Interfaces



WX_GUI

Sharing information between the team members



Gathering information about the source code

The screenshot shows the RefactorErl web interface. At the top, there is a navigation bar with tabs for QUERIES, DATABASE, ERRORS, DEPENDENCY GRAPH, and CODE DUPLICATES. The current page is displaying the results of a query for the symbol `@fun.references` in the file `mnesia_monitor:patch_env(...)`. The interface includes a search bar, a "Collapse all" button, and a "Database browser" section with a "See filesystem" button and a "Filename filter" input. The database browser shows a tree view with "kernel-2.16.1" and "mnesia-4.8" folders. The main content area displays the source code for `mnesia.erl`, with line numbers 221 through 251 visible. The code includes functions like `patched_start`, `stop`, and `change_config`.

localhost:8001/#/queries?file=%2Fusr%2Flib%2Fferlang%2Flib%2Fmnesia-4.8%2Fsrc%2Fmnesia.erl&id=z&pos=6415,644

Google

Logged in as melinda LOGOUT

RefactorErl @fun.references FROM mnesia_monitor:patch_env(...) Execute Queue 0/0

Query results

Collapse all Expand all

▼ mnesia_monitor:patch_env/2

- mnesia_monitor:patch_env(Env, Val)
- mnesia_monitor:patch_env(debug, Level)
- patch_env/2

Database browser See filesystem

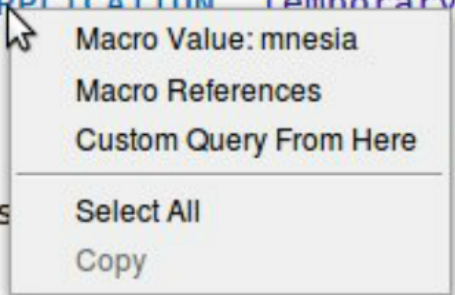
Show only erlang source files

Filename filter

- ▶ kernel-2.16.1
- ▶ mnesia-4.8

```
mnesia.erl
221     {error, {badarg, ExtraEnv}}.
222
223     patched_start([_Head | Tail]) when is_atom(Env) ->
224     case mnesia_monitor:patch_env(Env, Val) of
225     {error, Reason} ->
226     {error, Reason};
227     _NewVal ->
228     patched_start(Tail)
229     end;
230     patched_start([_Head | _]) ->
231     {error, {bad_type, Head}};
232     patched_start([]) ->
233     start().
234
235     stop() ->
236     case application:stop(?APPLICATION) of
237     ok -> stopped;
238     {error, {not_started, ?APPLICATION}} -> stopped;
239     Other -> Other
240     end.
241
242     change_config(extra_db_nodes, Ns) when is_list(Ns) ->
243     mnesia_controller:connect_nodes(Ns);
244     change_config(dc_dump_limit, N) when is_number(N), N > 0 ->
245     case mnesia_lib:is_running() of
246     yes ->
247     mnesia_lib:set(dc_dump_limit, N),
248     {ok, N};
249     _ ->
250     {error, {not_started, ?APPLICATION}}
251     end.
```

```
start() ->
  {Time , Res} = timer:tc(application, start, [?APPLICATION temporary]),
  Secs = Time div 1000000,
  case Res of
  ok ->
    verbose("Mnesia started, ~p seconds~n",[ Secs]),
    ok;
  {error, {already_started, mnesia}} ->
    verbose("Mnesia already started, ~p seconds~n",[ Secs]),
    ok;
  {error, R} ->
    verbose("Mnesia failed to start, ~p seconds: ~p~n",[ Secs, R]),
    {error, R}
  end.
```



- Macro Value: mnesia
- Macro References
- Custom Query From Here
- Select All
- Copy

Save as skeleton

Run

Query History ▾



Deselect Node

Choose previous query ▾

Current file: /usr/local/lib/erlang/lib/mnesia-4.7.1/src/mnesia.erl

Previous Queries Running Queries File Browser

Skeletons Last Result

File Browser

Filter:

- usr/local/lib/erlang/lib
 - kernel-2.15.1/include
 - mnesia-4.7.1/src
 - mnesia.erl**
 - mnesia.hrl
 - mnesia_backup.erl
 - mnesia_bup.erl
 - mnesia_checkpoint.erl
 - mnesia_checkpoint_sup.erl
 - mnesia_controller.erl
 - mnesia_dumper.erl
 - mnesia_event.erl
 - mnesia_frag.erl
 - mnesia_frag_hash.erl
 - mnesia_frag_old_hash.erl
 - mnesia_index.erl
 - mnesia_kernel_sup.erl
 - mnesia_late_loader.erl
 - mnesia_lib.erl
 - mnesia_loader.erl
 - mnesia_locker.erl
 - mnesia_log.erl
 - mnesia_monitor.erl
 - mnesia_recover.erl
 - mnesia_registry.erl
 - mnesia_schema.erl
 - mnesia_snmp_hook.erl
 - mnesia_snmp_sup.erl
 - mnesia_sp.erl
 - mnesia_subscr.erl
 - mnesia_sup.erl

```
149     end.
150
151 is_dollar_digits(Var) ->
152   case atom_to_list(Var) of
153     [$ | Digs] ->
154       is_digits(Digs);
155     _ ->
156       false
157   end.
158
159 is_digits([Dig | Tail]) ->
160   if
161     $0 <= Dig, Dig <= $9 ->
162       is_digits(Tail);
163     true ->
164       true
165   end;
166 is_digit(_) ->
167   true
168
169 has_var(X) when is_atom(X) ->
170   if
171     X == '_' ->
172       true;
173     is_atom(X) ->
174       is_dollar_digits(X);
175     true ->
176       false
177   end;
178 has_var(X) when is_tuple(X) ->
179   e_has_var(X, tuple_size(X));
180 has_var([H|T]) ->
181   case has_var(H) of
182     false -> has_var(T);
183     Other -> Other
184   end;
185 has_var(_) -> false.
186
187 e_has_var(_, 0) -> false;
188 e_has_var(X, Pos) ->
189   case has_var(element(Pos, X)) of
190     false -> e_has_var(X, Pos-1);
191     Other -> Other
192   end.
193
194 %%% Start and stop
195
```

Function References
Function Definitions

Run

Start investigation

Investigations

RefactorErl

Queries Files Errors Dependency Graphs Code Duplicates Investigations Log out viki

Unnamed

Starting function (m:f/a):
mnesia:start/1

Saved investigations

```
Options Hide Window1
212
213 start(ExtraEnv) when is_list(ExtraEnv) ->
214   case mnesia_lib:ensure_loaded(?APPLICATION) of
215     ok ->
216       patched_start(ExtraEnv);
217     Error ->
218       Error
219   end
```

```
Options Hide Window2
222 patched_start([Env, Val] | Tail) when is_atom(Env)
223 ->
224   case mnesia_monitor:patch_env(Env, Val) of
225     {error, Reason} ->
226       {error, Reason};
227     _NewVal ->
228       patched_start(Tail)
229   end
```

```
Options Hide Window3
27
28 check_fallback_dir_arg(Master, FA) ->
29   case FA#fallback_args.use_default_dir of
30     true ->
31       mnesia_lib:dir();
32     false when FA#fallback_args.scope == local ->
33       Dir = FA#fallback_args.mnesia_dir,
34       case catch mnesia_monitor:do_check_type(dir, Dir) of
35         {'EXIT', _R} ->
36           Reason = {badarg, {dir, Dir}, node()},
37           local_fallback_error(Master, Reason);
38         AbsDir ->
39           AbsDir
40       end;
41     false when FA#fallback_args.scope == global ->
42       Reason = {combine_error, global, dir, node()},
43       local_fallback_error(Master, Reason)
44   end
```

```
Options Hide Window4
41
42 do_check_type(access_module, A) when is_atom(A) -> A
```

mnesia_monitor:patch_env
@expr.funs

mnesia_monitor:do_check_type
@expr.funs



Options Hide Window1

```

212
213 start(ExtraEnv) when is_list(ExtraEnv) ->
214     case mnesia_lib:ensure_loaded(?APPLICATION) of
215     ok ->
216         patched_start(ExtraEnv);
217     Error ->
218         Error
219     end

```

pa
@

mnesia_monitor:patch
@expr.funs

Options Hide Window3



Dependency Graphs

Code Duplicates

Investigations

Log out

```
Options Hide Window2
222 patched_start([Env, Val] | Tail) when is_atom(Env)
223 >
224     case mnesia_monitor:patch_env(Env, Val) of
225     {error, Reason} ->
226         {error, Reason};
227     _NewVal ->
228         patched_start(Tail)
229     end
```

mnesia_monitor:patch_env
@expr.funs

Options Hide Window3

```
$27  
$28 check_fallback_dir_arg(Master, FA) ->  
$29     case FA#fallback_args.use_default_dir of  
$30     true ->  
$31         mnesia_lib:dir();  
$32     false when FA#fallback_args.scope == local ->  
$33         Dir = FA#fallback_args.mnesia_dir,  
$34         case catch mnesia_monitor:do_check_type(dir, Dir) of  
$35             {'EXIT', _R} ->  
$36                 Reason = {badarg, {dir, Dir}, node()},  
$37                 local_fallback_error(Master, Reason);  
$38             AbsDir ->  
$39                 AbsDir  
$40         end;  
$41     false when FA#fallback_args.scope == global ->  
$42         Reason = {combine_error, global, dir, node()},  
$43         local_fallback_error(Master, Reason)  
$44 end
```

mnesia_monitor:do_check_type
@expr.funs

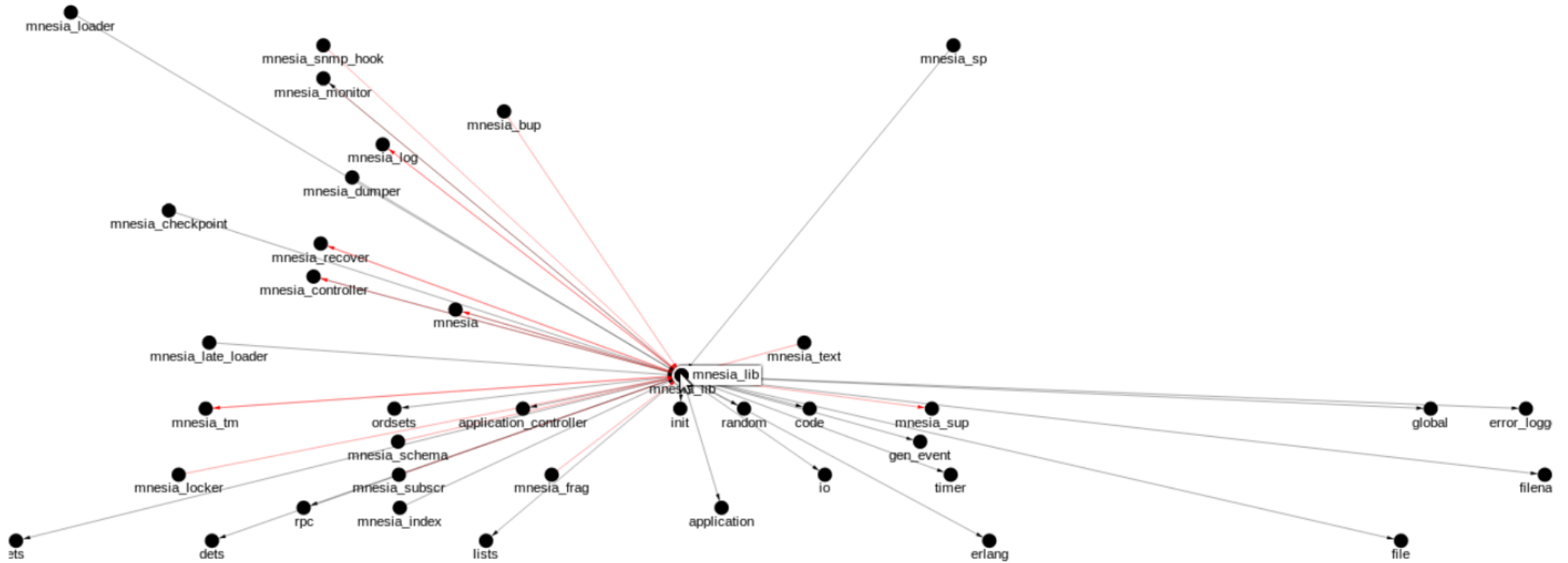
@expr.fun

W

Options Hide Window4

```
41  
42do_check_type(access_module, A) when is_atom(A) -> A
```

Dependencies



Industrial Applications

Network Simulators

Telecom Gateway Controller

AXD ATM switch

More than 4 millions of LOC

Code Comprehension

Clustering

Industrial Applications

Network Simulators

Telecom Gateway Controller

ATM switch

More than 4 mill

More than 4 millions of

Code Comprehension

Clustering

Experience

"A problem solved in one hour
using the query language"

"Without the tool: approx. 1 day"

Easy to setup and use

Initial setup - 2-3 hours

Build the database - few hours

- only once at the beginning

Use the tool!

Use the tool!

Some advantages

shorten time-consuming daily jobs

make the possibility of better teamwork in different ways

reduce human faults

ease deploying releases

minimise the training time of newbies

refactorerl.com

pnyf.inf.elte.hu/trac/refactorerl

News



Pattern Discovery

<http://paraphrase-enlarged.elte.hu/>

Demo time

<http://plc.inf.elte.hu/erlang/cmd.txt>

Demo time

<http://plc.inf.elte.hu/erlang/cmd.txt>

Quick Startup

```
bin/referl -build tool  
bin/referl -db kcmuni
```

```
ri:q(mods).  
ri:q("mods.funs").  
ri:q("mods[name~mnesia_snmp].funs").
```

```
ri:ls().
```

```
ri:envs().  
ri:addenv(appbase, "/home/melinda/mylibpath").  
ri:add(usr, mnesia).
```

```
ri:start_web2().
```

Simple easy to use features

Code Browser

Click and Browse

- function, record, record_field, variable, macro, etc
- built-in dynamic function calls
- references, definition, macro value on right click

Synchronised function list

Search and find:

- plain text search
- semantic search

Dependence graphs

Database & File System Browser

Search and view Erlang & other files

Advanced Features

Semantic Queries & Parametrised Queries

- Collaborative work support
- query & graph sharing
 - stateful links

```
mods[name=mnesia_log]
  .funs.exprs
    .sub[type=tuple[
      .sub[index=1]
        .origin[type=atom,
          value =backup_args]]]
```

```
mods[name=mnesia_log]
  .funs[name=open_log]
    .refs[
      .param[index=1]
        .origin[type=atom,
          value=decision_log]]]
```

Investigations

s & Parametrised Queries

```
mods[name=mnesia_log]
  .funcs.exprs
    .sub[type=tuple[
      .sub[index=1]
        .origin[type=atom,
          value =backup_args]]]
```

ort
ing

```
mods[name=mnesia_log]
  .funcs[name=open_log]
    .refs[
      .param[index=1]
```

```
.sub[index=1]
  .origin[type=atom,
    value =backup_args]]]
```

```
mods[name=mnesia_log]
  .funcs[name=open_log]
    .refs[
      .param[index=1]
        .origin[type=atom,
          value=decision_log]]]
```

And a bit more advanced stuff...

Centralised management support

- restricted mode for users
- web or console based administration

```
Quick Start
Example: install
Example: uninstall

#sig
#name
#url=http://www.telabsystems.com
#url=http://www.telabsystems.com

#sig
#url=http://www.telabsystems.com
#url=http://www.telabsystems.com

#sig
#url=http://www.telabsystems.com
#url=http://www.telabsystems.com
```

Q & A

Q & A