Refactoring Erlang with Wrangler

Huiqing Li
Simon Thompson

School of Computing
University of Kent

ProTest
property based testing

University of Kent
Computing

Overview

Refactoring.

Tools and tool building.

Clone detection.

Refactoring and testing.

Tool demo … Huiqing, George and Simon.

Introduction

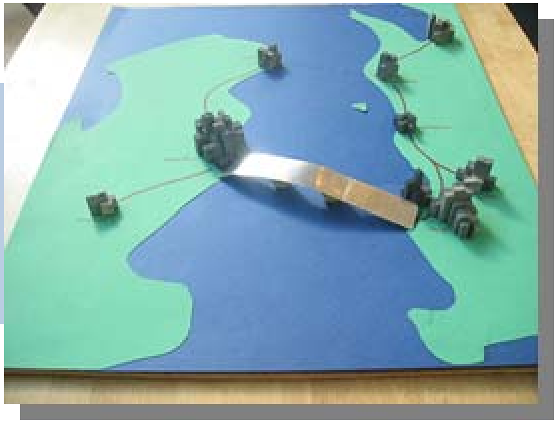
Design

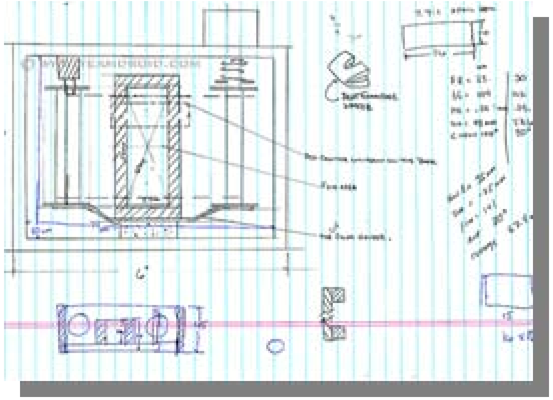
Models

Prototypes

Design documents

Visible artifacts





All in the code

Functional programs embody their design in their code.

Successful programs evolve … as do their tests, makefiles etc.

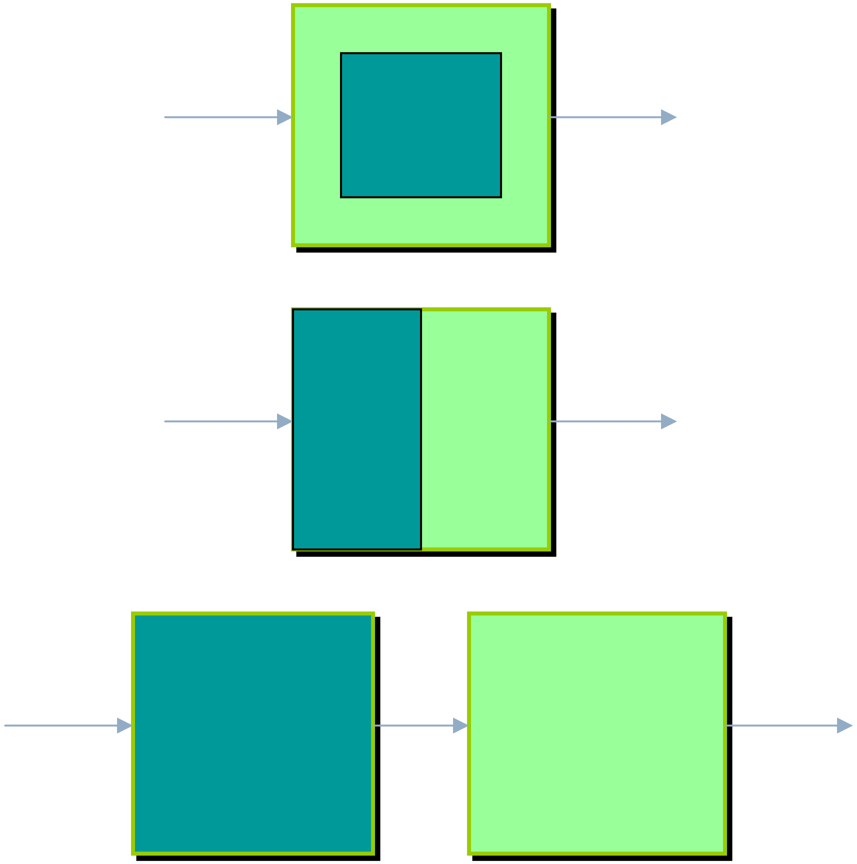
```
loop(Frequencies) ->
  receive
    {request, Pid, allocate} ->
      {NewFrequencies, Reply} =
allocate(Frequencies, Pid),
      reply(Pid, Reply),
      loop(NewFrequencies);
    {request, Pid , {deallocate, Freq}} ->
      NewFrequencies=deallocate(Frequencies,
Freq),
      reply(Pid, ok),
      loop(NewFrequencies);
    {'EXIT', Pid, _Reason} ->
      NewFrequencies = exited(Frequencies, Pid),
      loop(NewFrequencies);
    {request, Pid, stop} ->
      reply(Pid, ok)
  end.

exited({Free, Allocated}, Pid) ->
  case lists:keysearch(Pid,2,Allocated) of
    {value,{Freq,Pid}} ->
       NewAllocated =
lists:keydelete(Freq,1,Allocated),
      {[Freq|Free],NewAllocated};
    false ->
      {Free,Allocated}
  end.
```

Soft-Ware

There's no single correct design …

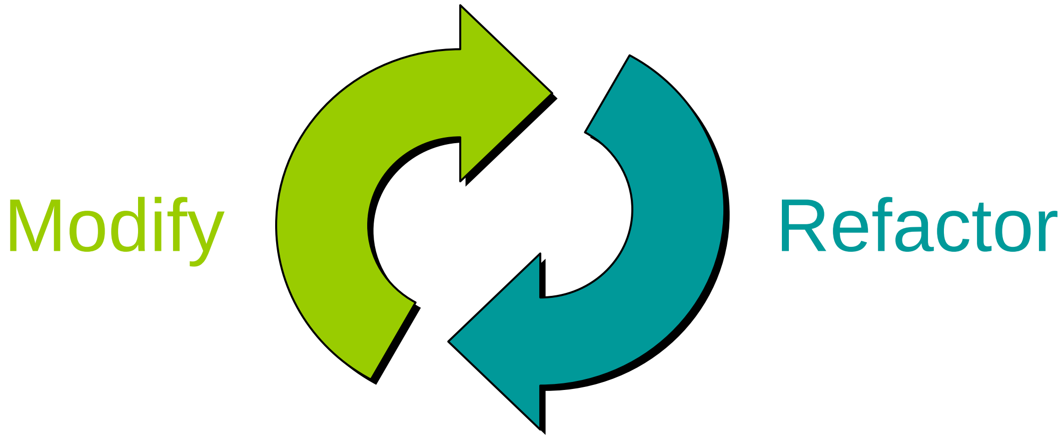
… different options for different situations.

Maintain flexibility as the system evolves.



Refactoring

Refactoring means changing the design or structure of a program … without changing its behaviour.



Not just programming

Paper or presentation

- moving sections about; amalgamate sections; move inline code to a figure; animation.

Proof

- add lemma; remove, amalgamate hypotheses.

Tests

- refactor tests themselves, or evolve them in synch with the program.

Generalisation and renaming

```
-module (test).
-export([f/1]).

add_one ([H|T]) ->
  [H+1 | add_one(T)];

add_one ([]) -> [].

f(X) -> add_one(X).
```

```
-module (test).
-export([f/1]).

add_int (N, [H|T]) ->
 [H+N | add_int(N,T)];

add_int (N,[]) -> [].

f(X) -> add_int(1, X).
```

Generalisation

```
-export([printList/1]).

printList([H|T]) ->
  io:format("~p\n",[H]),
  printList(T);
printList([]) -> true.


printList([1,2,3])
```

```
-export([printList/2]).

printList(F,[H|T]) ->
  F(H),
  printList(F, T);
printList(F,[]) -> true.


printList(
  fun(H) ->
   io:format("~p\n", [H])
  end,
  [1,2,3]).
```

The tool

Refactoring tool support

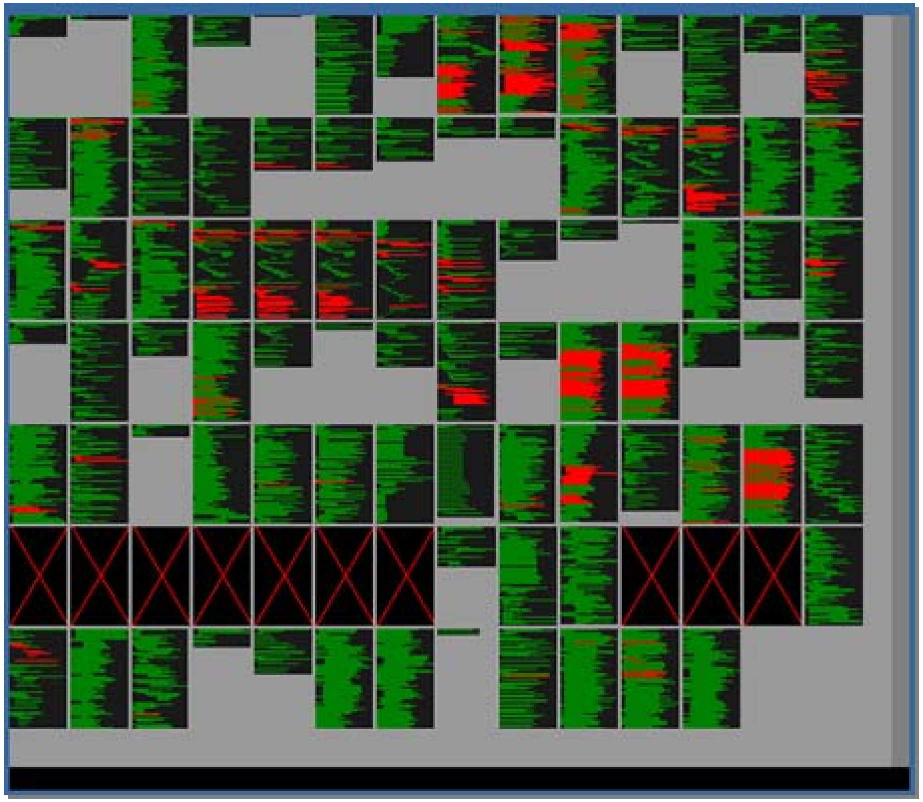
Bureaucratic and diffuse.

Tedious and error prone.

Semantics: scopes, types, modules, …

Undo/redo

Enhanced creativity



Wrangler

Refactoring tool for Erlang

Integrated into Emacs and Eclipse

Multiple modules

Structural, process, macro refactorings

Duplicate code detection …

… and elimination

Testing / refactoring

"Similar" code identification

Property discovery

Semantic analysis

Binding structure

- Dynamic atom creation, multiple binding occurrences, pattern semantics etc.

Module structure and projects

- No explicit projects for Erlang; cf Erlide / Emacs.

Type and effect information

- Need effect information for e.g. generalisation.

Erlang refactoring: challenges

Multiple binding occurrences of variables.

Indirect function call or function spawn:

```
        apply (lists, rev,  [[a,b,c]])
```

Multiple arities … multiple functions: `rev/1`

Concurrency

Refactoring within a design library: OTP.

Side-effects.

Static *vs* dynamic

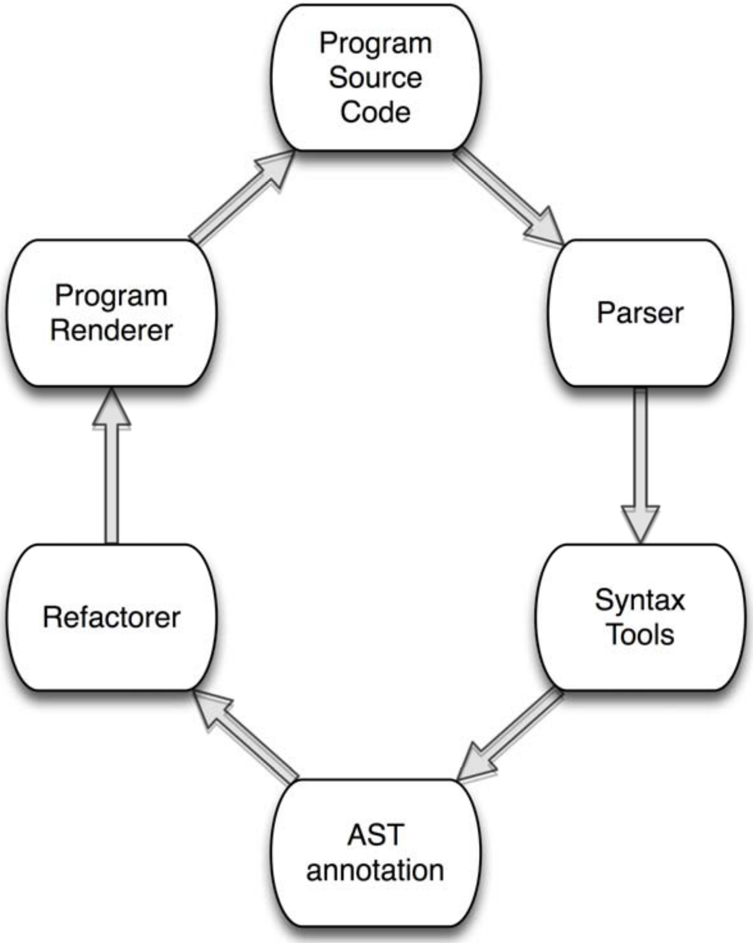
Aim to check conditions statically.

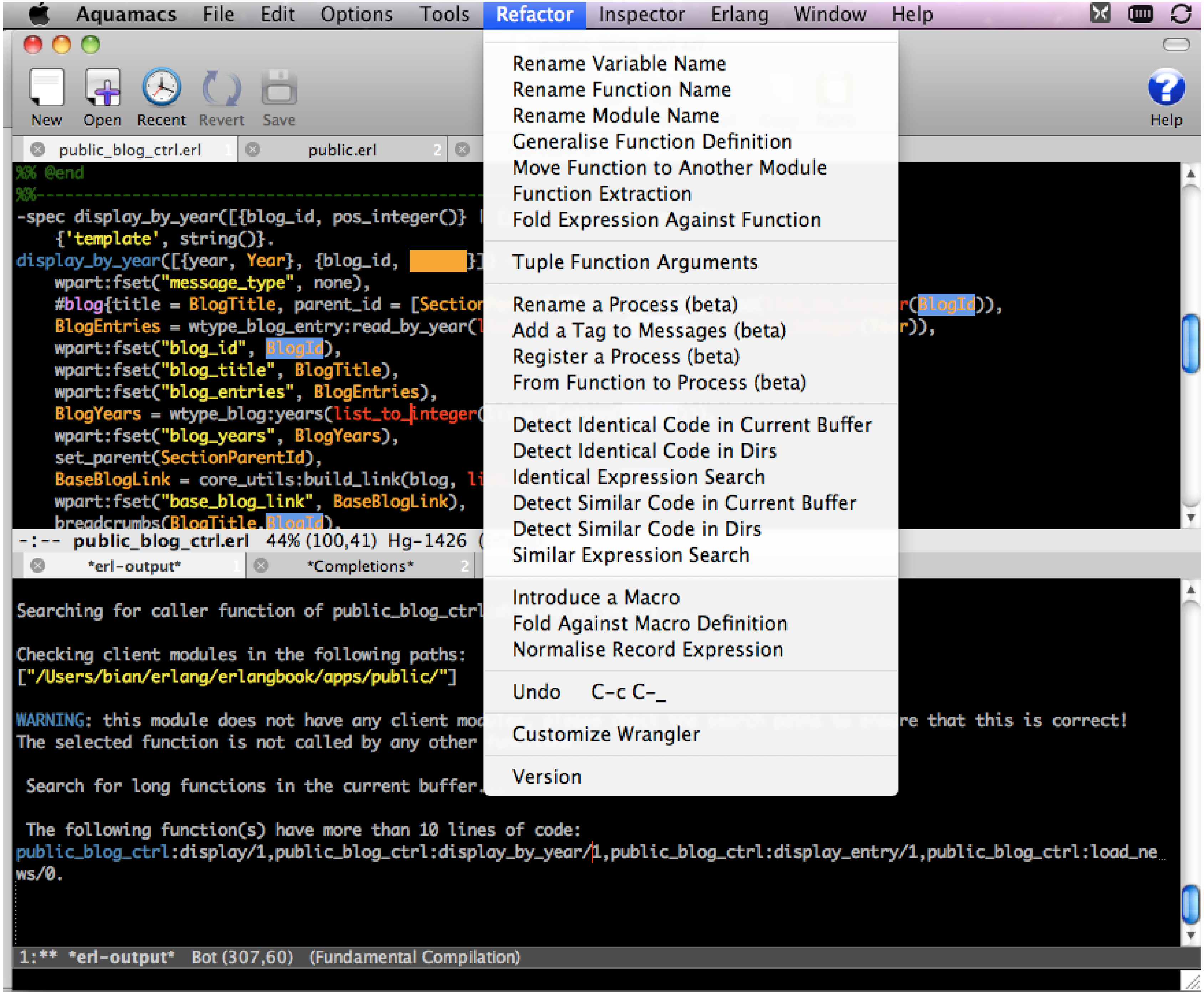
Static analysis tools possible … but some aspects intractable: e.g. dynamically manufactured atoms.

Conservative *vs* liberal.

Compensation?

Architecture of Wrangler




Aquamacs
File
Edit
Options
Tools
Refactor
Inspector
Erlang
Window
Help
New
Open
Recent
Revert
Save
Help
public_blog_ctrl.erl
public.erl
erl-output
Completions
Rename Variable Name
Rename Function Name
Rename Module Name
Generalise Function Definition
Move Function to Another Module
Function Extraction
Fold Expression Against Function
Tuple Function Arguments
Rename a Process (beta)
Add a Tag to Messages (beta)
Register a Process (beta)
From Function to Process (beta)
Detect Identical Code in Current Buffer
Detect Identical Code in Dirs
Identical Expression Search
Detect Similar Code in Current Buffer
Detect Similar Code in Dirs
Similar Expression Search
Introduce a Macro
Fold Against Macro Definition
Normalise Record Expression
Undo   C-c C-_
Customize Wrangler
Version

```
%% @end
%%-----------------------------------------
-spec display_by_year([{blog_id, pos_integer()} |
    {'template', string()}.
display_by_year([{year, Year}, {blog_id,       }])
    wpart:fset("message_type", none),
    #blog{title = BlogTitle, parent_id = [Section                r(BlogId)),
    BlogEntries = wtype_blog_entry:read_by_year(                 r)),
    wpart:fset("blog_id", BlogId),
    wpart:fset("blog_title", BlogTitle),
    wpart:fset("blog_entries", BlogEntries),
    BlogYears = wtype_blog:years(list_to_integer(
    wpart:fset("blog_years", BlogYears),
    set_parent(SectionParentId),
    BaseBlogLink = core_utils:build_link(blog, li
    wpart:fset("base_blog_link", BaseBlogLink),
    breadcrumbs(BlogTitle, BlogId),
-:-- public_blog_ctrl.erl   44% (100,41) Hg-1426 (
```

```
Searching for caller function of public_blog_ctrl

Checking client modules in the following paths:
["/Users/bian/erlang/erlangbook/apps/public/"]

WARNING: this module does not have any client mod                re that this is correct!
The selected function is not called by any other

 Search for long functions in the current buffer.

 The following function(s) have more than 10 lines of code:
public_blog_ctrl:display/1,public_blog_ctrl:display_by_year/1,public_blog_ctrl:display_entry/1,public_blog_ctrl:load_ne
ws/0.

1:** *erl-output*  Bot (307,60)  (Fundamental Compilation)
```

Refactorings in Wrangler

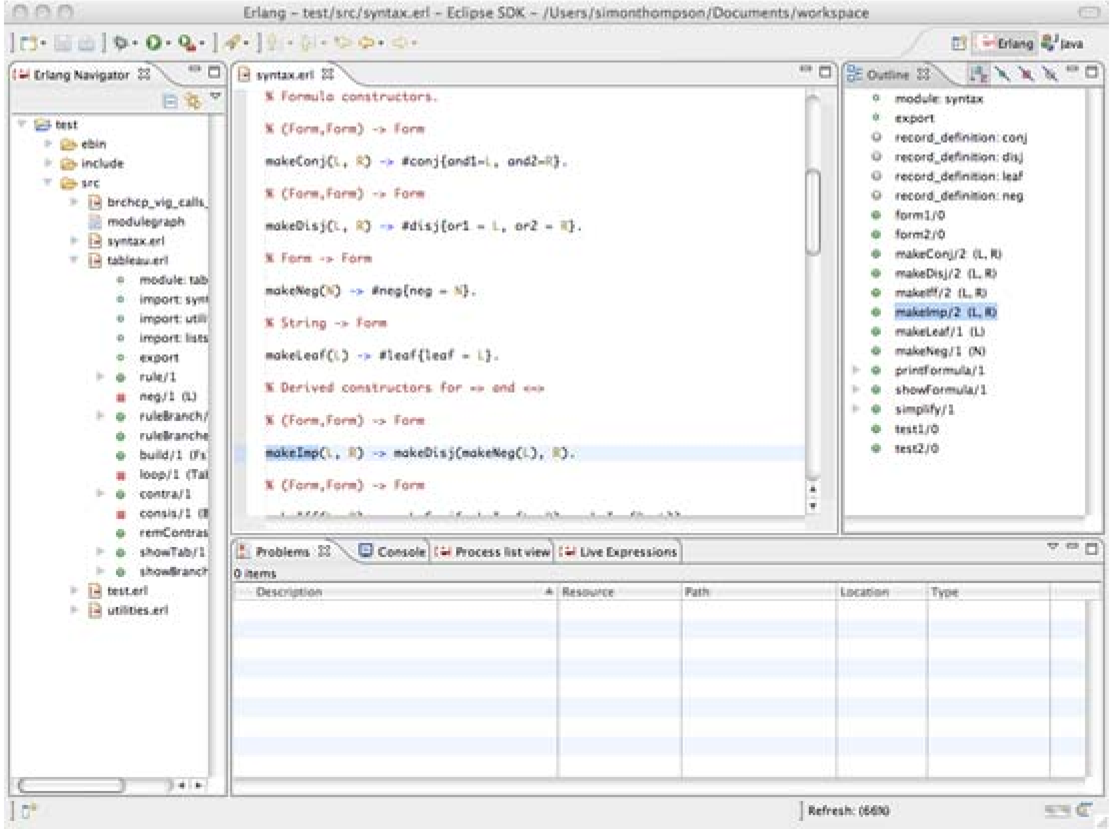
- Renaming variable, function, module, process
- Function generalisation
- Move function between modules.
- Function extraction
- Fold against definition
- Introduce and fold against macros.
- Tuple function arguments together
- Register a process
- From function to process
- Add a tag to messages

All these refactorings work across multiple-module projects and respect macro definitions.

Integration with ErlIDE

Tighter control of what's a project.

Potential for adoption by newcomers to the Erlang community.



Erlang – test/src/syntax.erl – Eclipse SDK – /Users/simonthompson/Documents/workspace

```
% Formula constructors.
% (Form,Form) -> Form
makeConj(L, R) -> #conj{and1=L, and2=R}.
% (Form,Form) -> Form
makeDisj(L, R) -> #disj{or1 = L, or2 = R}.
% Form -> Form
makeNeg(N) -> #neg{neg = N}.
% String -> Form
makeLeaf(L) -> #leaf{leaf = L}.
% Derived constructors for => and <=>
% (Form,Form) -> Form
makeImp(L, R) -> makeDisj(makeNeg(L), R).
% (Form,Form) -> Form
```

ProTest property based testing

University of Kent Computing

Clone detection

Clone detection

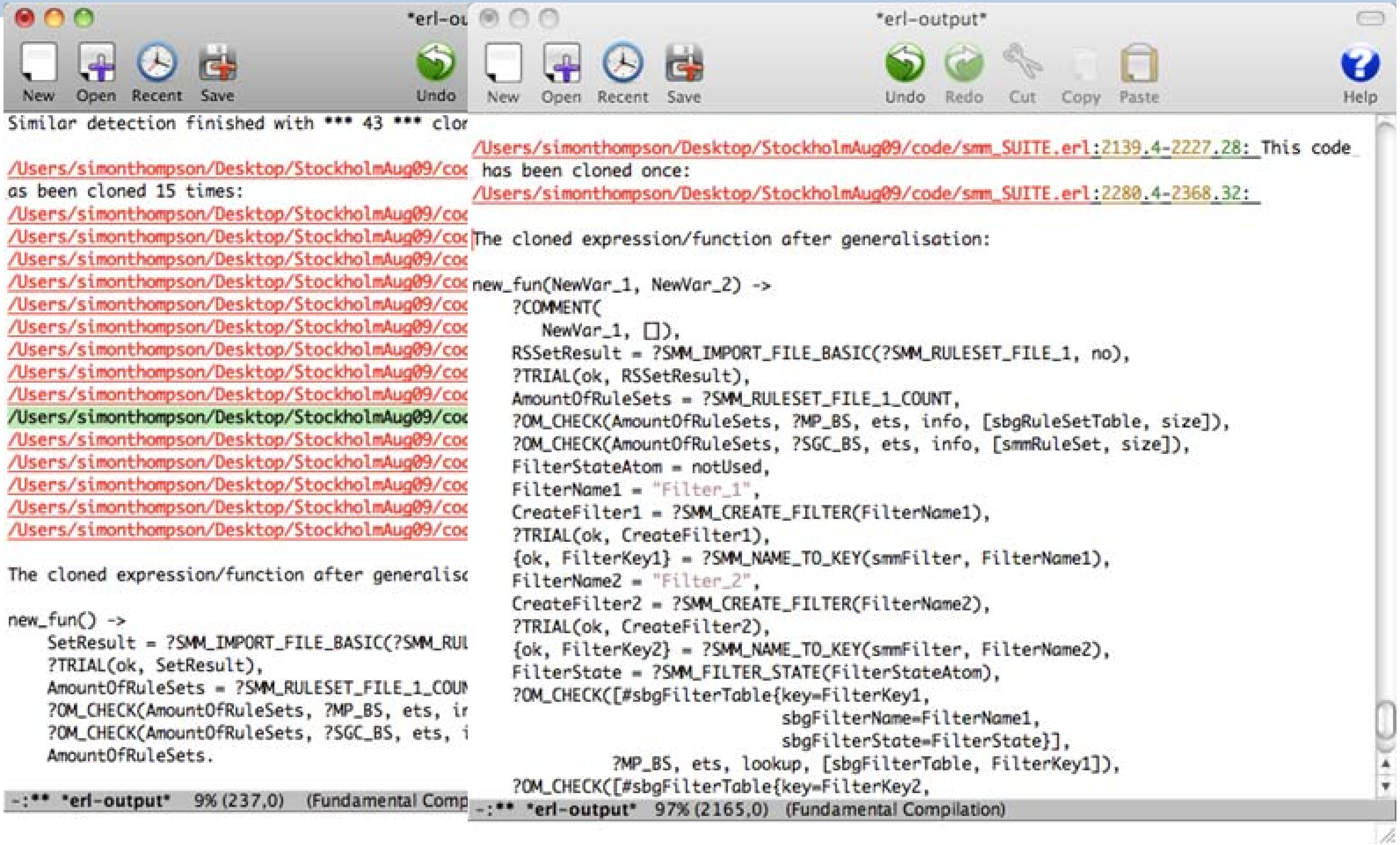
The Wrangler clone detector

- Relatively efficient
- No false positives

Refactorings support interactive removal of clones.

Integrated in the development environment.

Clone detection



```
Similar detection finished with *** 43 *** clon

/Users/simonthompson/Desktop/StockholmAug09/coc
as been cloned 15 times:
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc
/Users/simonthompson/Desktop/StockholmAug09/coc

The cloned expression/function after generalisc

new_fun() ->
    SetResult = ?SMM_IMPORT_FILE_BASIC(?SMM_RUL
    ?TRIAL(ok, SetResult),
    AmountOfRuleSets = ?SMM_RULESET_FILE_1_COUN
    ?OM_CHECK(AmountOfRuleSets, ?MP_BS, ets, ir
    ?OM_CHECK(AmountOfRuleSets, ?SGC_BS, ets, i
    AmountOfRuleSets.

-:** *erl-output*   9% (237,0)   (Fundamental Comp
```

```
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:2139.4-2227.28: This code
 has been cloned once:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:2280.4-2368.32:

The cloned expression/function after generalisation:

new_fun(NewVar_1, NewVar_2) ->
    ?COMMENT(
       NewVar_1, []),
    RSSetResult = ?SMM_IMPORT_FILE_BASIC(?SMM_RULESET_FILE_1, no),
    ?TRIAL(ok, RSSetResult),
    AmountOfRuleSets = ?SMM_RULESET_FILE_1_COUNT,
    ?OM_CHECK(AmountOfRuleSets, ?MP_BS, ets, info, [sbgRuleSetTable, size]),
    ?OM_CHECK(AmountOfRuleSets, ?SGC_BS, ets, info, [smmRuleSet, size]),
    FilterStateAtom = notUsed,
    FilterName1 = "Filter_1",
    CreateFilter1 = ?SMM_CREATE_FILTER(FilterName1),
    ?TRIAL(ok, CreateFilter1),
    {ok, FilterKey1} = ?SMM_NAME_TO_KEY(smmFilter, FilterName1),
    FilterName2 = "Filter_2",
    CreateFilter2 = ?SMM_CREATE_FILTER(FilterName2),
    ?TRIAL(ok, CreateFilter2),
    {ok, FilterKey2} = ?SMM_NAME_TO_KEY(smmFilter, FilterName2),
    FilterState = ?SMM_FILTER_STATE(FilterStateAtom),
    ?OM_CHECK([#sbgFilterTable{key=FilterKey1,
                               sbgFilterName=FilterName1,
                               sbgFilterState=FilterState}],
              ?MP_BS, ets, lookup, [sbgFilterTable, FilterKey1]),
    ?OM_CHECK([#sbgFilterTable{key=FilterKey2,
-:** *erl-output*   97% (2165,0)   (Fundamental Compilation)
```

Clone detection

Clone detection …

… and elimination.

Find code that is similar …

… common abstraction …

Examples:

Test code from Ericsson: different medium and codec.

Clone removal example: 2.6k to 2.0k and counting.

Property extraction

Fitting into the ProTest project: move from test cases to properties in QuickCheck.

Use Wrangler to spot clones, and to build properties from them.

Support property extraction from 'free' and EUnit tests.

Refactoring and tests

Respecting test code in EUnit, QuickCheck and Common Test.

Refactor tests along with code refactoring.

Refactor tests: e.g.

- Convert tests into EUnit tests.
- Group EUnit tests into a single test generator.
- Move EUnit tests into a separate test module.
- Normalise EUnit tests.
- Extract common setup and tear-down code into EUnit fixtures.

Interface and user experience

User experience: preview changes, code inspector, …

Multi-version: Erlang, OS, Java, Eclipse.

Further integration into Erlide: allow use of the contextual menu.

Windows installer.

Hands-on

Installation: Mac OS X and Linux

Requires: Erlang release R11B-5, 12B or 13B

Installation: Mac OS X and Linux

Download Wrangler from

http://www.cs.kent.ac.uk/projects/wrangler/

or get it from the memory stick …

In the wrangler directory

```
./configure

make

sudo make install
```

Installation: Mac OS X and Linux

Add to `~/.emacs` file:

```
(add-to-list 'load-path
        "/usr/local/share/wrangler/elisp")
(require 'wrangler)
```

If you’re installing emacs now, then you add the following lines to your `~/.emacs` file

```
(setq load-path (cons "/usr/local/otp/lib/tools-<ToolsVer>/emacs"
                     load-path))
(setq erlang-root-dir "/usr/local/otp")
(setq exec-path (cons "/usr/local/otp/bin" exec-path))
(require 'erlang-start)
```

Installation: Debian package

Will be available from the homepage in the next week …

… also on the memory stick.

Installation: Windows

Requires R11B-5, 12B, 13B + Emacs

Download installer from

http://www.cs.kent.ac.uk/projects/wrangler/

Requires no other actions.

Installation: Eclipse + ErlIDE

Requires Erlang R11B-5 or later, if it isn't already present on your system.

On Windows systems, use a path with no spaces in it.

Install Eclipse 3.4 or 3.5, if you didn't already.

All the details at

http://erlide.sourceforge.net/

Starting Wrangler in Emacs

Open emacs, and open a `.erl` file.

`M-x erlang-refactor-on` or ...

... `C-c, C-r`

New menus: Refactor and Inspector

Customise for dir

Undo `C-c, C-_`

Preview Feature

Preview changes before confirming the change

Emacs `ediff` is used.

Stopping Wrangler in Emacs

`M-x erlang-refactor-off` to stop Wrangler

Shortcut `C-c, C-r`

Carrying on …

Try on your own project code …

Feedback:

erlang-refactor@kent.ac.uk or

H.Li@kent.ac.uk