Improving your (test) code with Wrangler

Huiqing Li, Simon Thompson
University of Kent

Andreas Schumacher
Ericsson Software Research

Adam Lindberg
Erlang Training and Consulting

ProTest
property based testing

University of Kent | Computing

CODE CODE CODE

ProTest
property based testing

University of Kent | Computing

Overview

Refactoring.

The Wrangler tool.

Clone detection.

Why test code?

Case study of SIP message manipulation tests.

General lessons.

Introduction

“It’s all in the code, stupid”

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.
```

ProTest property based testing

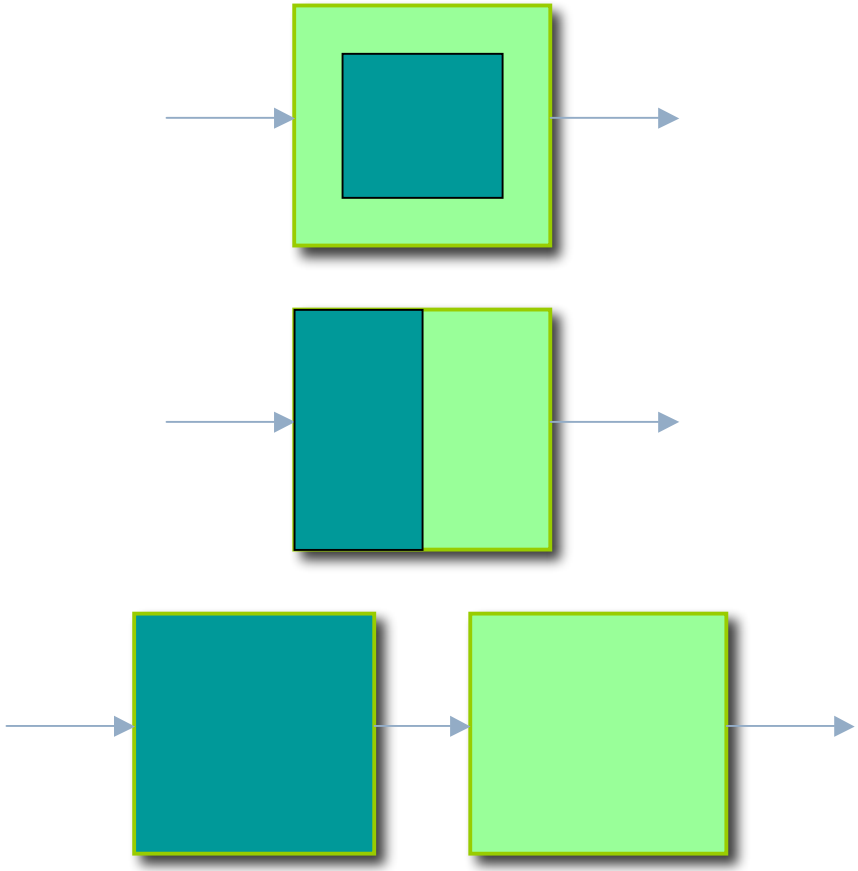
University of Kent Computing

Soft-Ware

There's no single correct design …

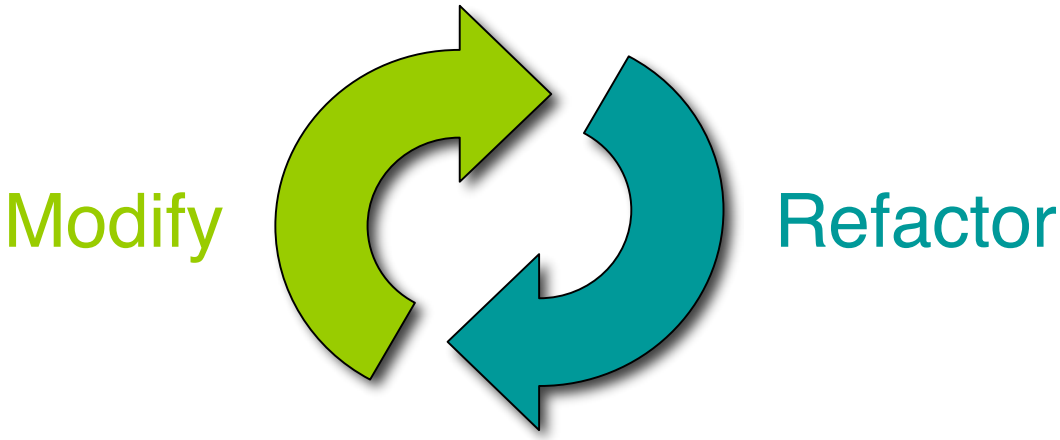
… different options for different situations.

Maintain flexibility as the system evolves.



Refactoring

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



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]).
```

ProTest property based testing

University of Kent Computing

The tool

Refactoring tool support

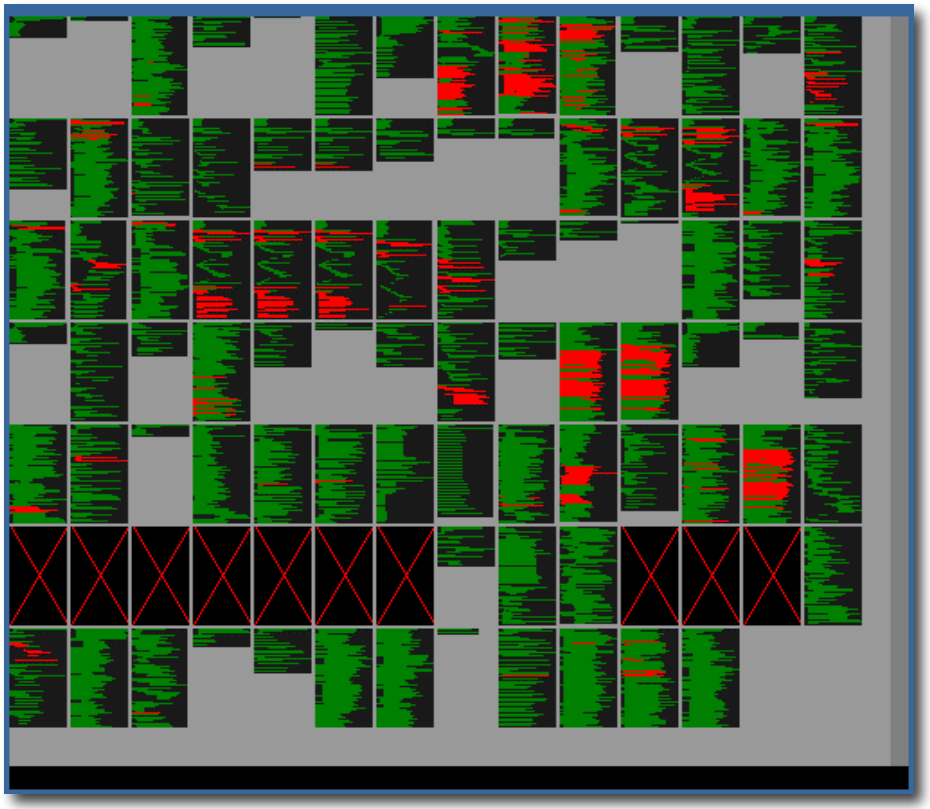
Bureaucratic and diffuse.

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

Static *vs* dynamic

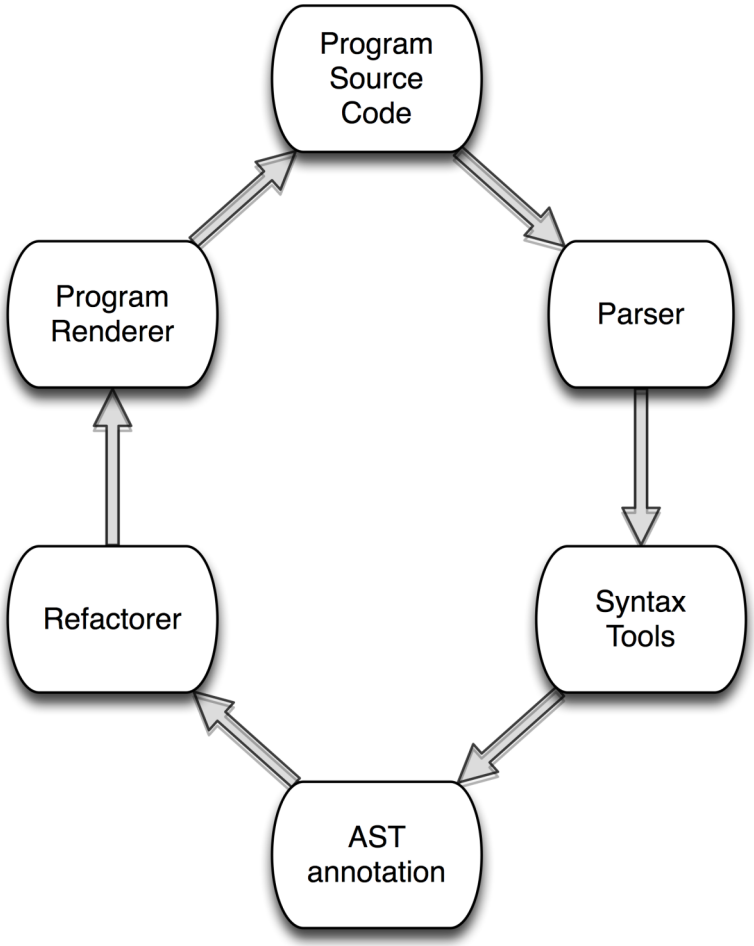
Aim to check conditions statically.

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

Conservative *vs* liberal.

Compensation?

Architecture of Wrangler



ProTest
property based testing

University of Kent | Computing

Aquamacs File Edit Options Tools Refactor Inspector Erlang Window Help

New Open Recent Revert Save Help

public_blog_ctrl.erl public.erl

```
%% @end
%%----------------------------------------------------------------
-spec display_by_year([{blog_id, pos_integer()} |
    {'template', string()}.
display_by_year([{year, Year}, {blog_id, [redacted]}]
    wpart:fset("message_type", none),
    #blog{title = BlogTitle, parent_id = [Section
    BlogEntries = wtype_blog_entry:read_by_year(
    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

erl-output *Completions*

```
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
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)

- 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-_

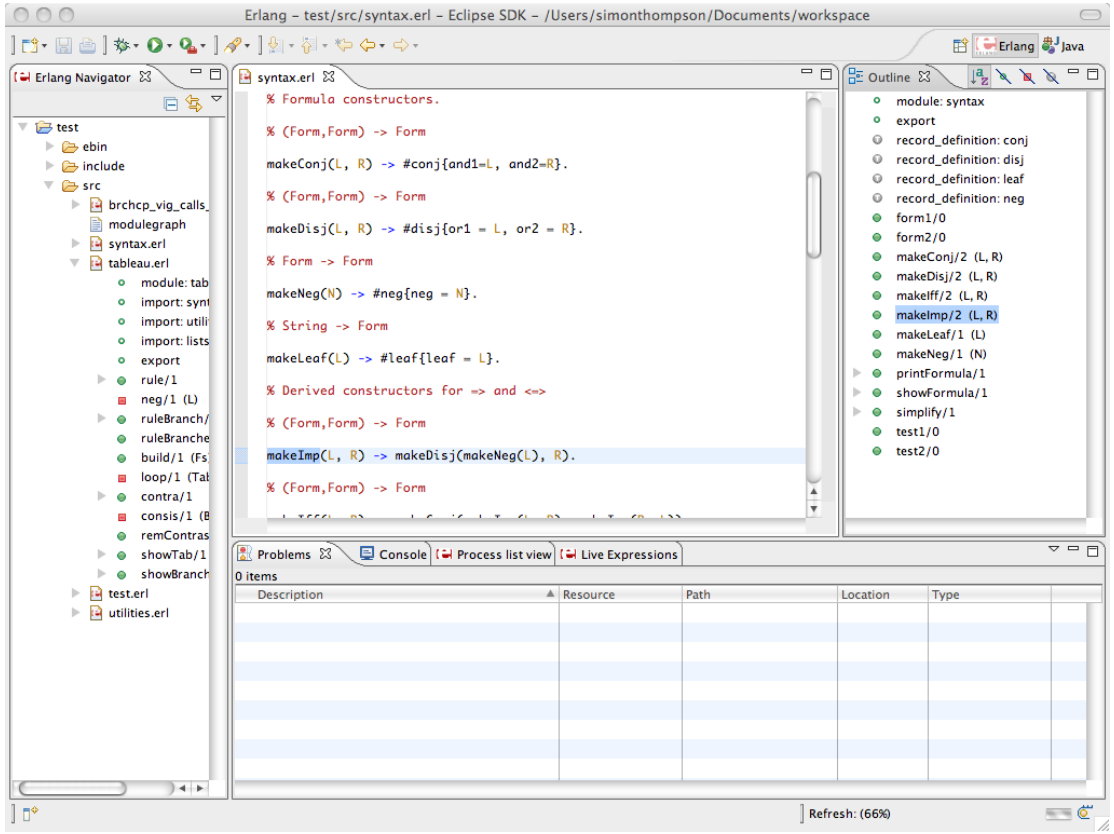
- Customize Wrangler

- Version

Integration with ErlIDE

Tighter control of what's a project.

Potential for adoption by newcomers to the Erlang community.



ProTest property based testing

University of Kent Computing

Clone detection

‘Code smells’

Bad smell … time to refactor?

- Name does not reflect the meaning
- Function too long
- Code not actually used
- Bad module structure
- Excessive nesting
- Duplicated code

Duplicate code considered harmful

- Increases the probability of bug propagation.
- Increases the size of the source code and the executable.
- Increases compile time.
- Increases the cost of maintenance.

But it's not *always* a problem …

Clone detection

The Wrangler clone detector

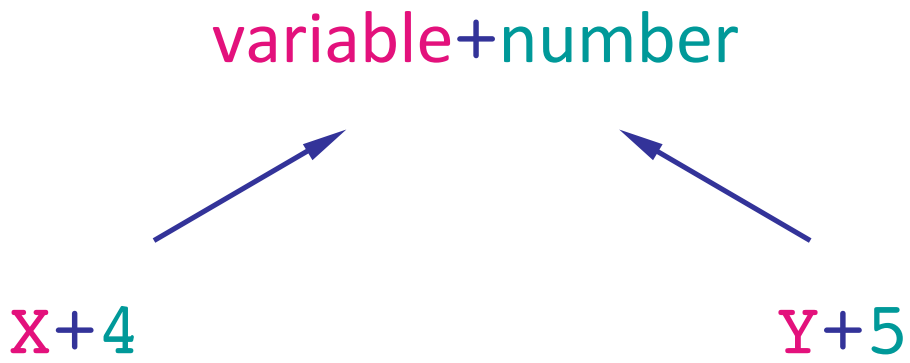
- relatively efficient
- no false positives

Interactive removal of clones …

… under user guidance.

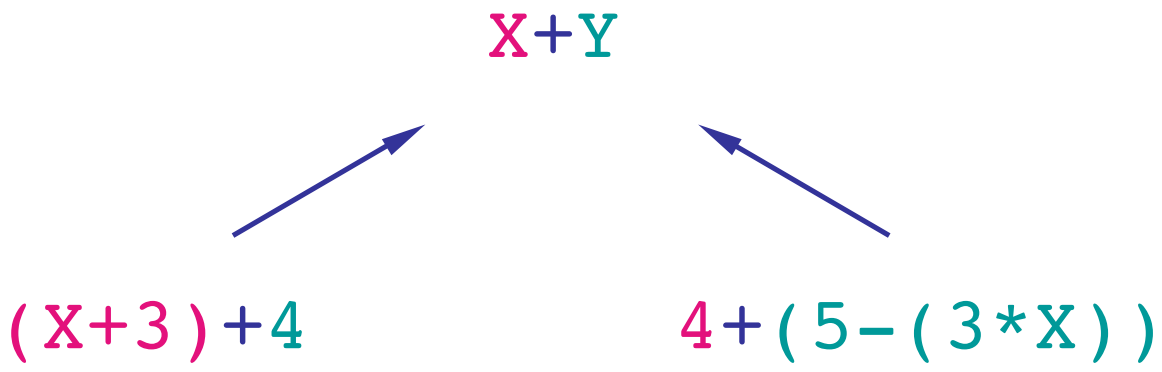
Integrated into the development environment.

What is ‘identical’ code?



Identical if values of literals and variables ignored, but respecting binding structure.

What is ‘similar’ code?



The anti-unification gives the (most specific) common generalisation.

Detection	Expression search
All clones in a project meeting the threshold parameters …	All instances similar to this expression …
… and their common generalisations.	… and their common generalisation.
Default threshold: ≥ 5 expressions and similarity of ≥ 0.8.	Default threshold: ≥ 20 tokens.

SIP Case Study

Why test code particularly?

Many people touch the code.

Write some tests … write more by copy, paste and modify.

Similarly with long-standing projects, with a large element of legacy code.

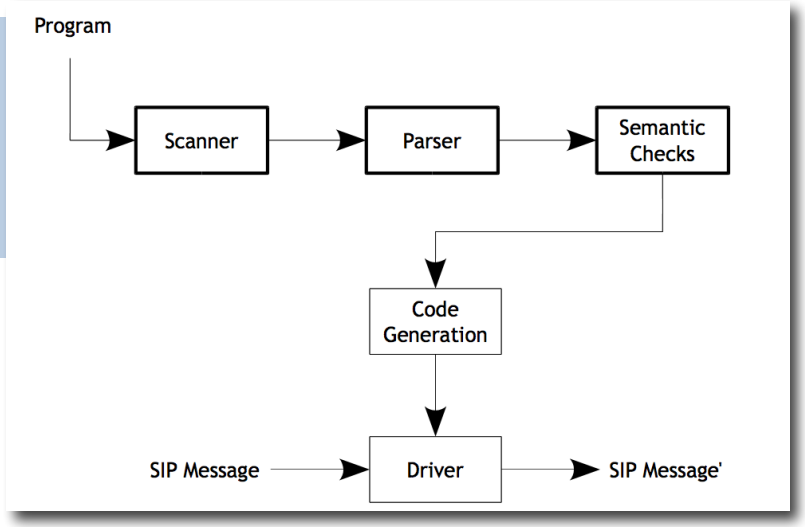
“Who you gonna call?”

Can reduce by 20% just by aggressively removing all the clones identified …

… what results is of no value at all.

Need to call in the domain experts.

SIP case study



Session Initiation Protocol

SIP message processing allows rewriting rules to transform messages.

SIP message manipulation (SMM) is tested by `smm_SUITE.erl`, 2658 LOC.

Reducing the case study

1	2658	6	2218	11	2131
2	2342	7	2203	12	2097
3	2231	8	2201	13	2042
4	2217	9	2183	…	…
5	2216	10	2149		

Step 1

The largest clone class has 15 members.

The suggested function has no parameters, so the code is literally repeated.

erl-output

New Open Recent Save Undo Redo Cut Copy Paste Help

```
Similar detection finished with *** 43 *** clone(s) found.

/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:196.4-202.71: This code h
as been cloned 15 times:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:377.4-383.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:693.4-699.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:755.4-761.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:807.4-813.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:904.4-910.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:988.4-994.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:1084.4-1090.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:1497.4-1503.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:1585.4-1591.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:1719.4-1725.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:1803.4-1809.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:2026.4-2032.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:2143.4-2149.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:2284.4-2290.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:2428.4-2434.71:

The cloned expression/function after generalisation:

new_fun() ->
    SetResult = ?SMM_IMPORT_FILE_BASIC(?SMM_RULESET_FILE_1, no),
    ?TRIAL(ok, SetResult),
    AmountOfRuleSets = ?SMM_RULESET_FILE_1_COUNT,
    ?OM_CHECK(AmountOfRuleSets, ?MP_BS, ets, info, [sbgRuleSetTable, size]),
    ?OM_CHECK(AmountOfRuleSets, ?SGC_BS, ets, info, [smmRuleSet, size]),
    AmountOfRuleSets.
```

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

ProTest property based testing

University of Kent Computing

Not step 1

The largest clone has 88 lines, and 2 parameters.

But what does it represent?

What to call it?

Best to work bottom up.

erl-output

New Open Recent Save Undo Redo Cut Copy Paste Help

```
/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)
```

ProTest property based testing

University of Kent Computing

The general pattern

Identify a clone.

Introduce the corresponding generalisation.

Eliminate all the clone instances.

So what's the complication?

Step 3

23 line clone occurs; choose to replace a smaller clone.

Rename function and parameters, and reorder them.

```
new_fun() ->
  {FilterKey1, FilterName1, FilterState, FilterKey2,
   FilterName2} = create_filter_12(),
  ?OM_CHECK([#smmFilter{key=FilterKey1,
                filterName=FilterName1,
                filterState=FilterState,
                module=undefined}],
      ?SGC_BS, ets, lookup, [smmFilter, FilterKey1]),
  ?OM_CHECK([#smmFilter{key=FilterKey2,
                filterName=FilterName2,
                filterState=FilterState,
                module=undefined}],
      ?SGC_BS, ets, lookup, [smmFilter, FilterKey2]),
  ?OM_CHECK([#sbgFilterTable{key=FilterKey1,
                sbgFilterName=FilterName1,
                sbgFilterState=FilterState}],
      ?MP_BS, ets, lookup, [sbgFilterTable, FilterKey1]),
  ?OM_CHECK([#sbgFilterTable{key=FilterKey2,
                sbgFilterName=FilterName2,
```

```
check_filter_exists_in_sbgFilterTable(FilterKey, FilterName, FilterState) ->
  ?OM_CHECK([#sbgFilterTable{key=FilterKey,
                sbgFilterName=FilterName,
                sbgFilterState=FilterState}],
      ?MP_BS, ets, lookup, [sbgFilterTable, FilterKey]).
```

ProTest property based testing

University of Kent Computing

Steps 4, 5

2 variants of `check_filter_exists_in_sbgFilterTable` …

- Check for the filter occurring uniquely in the table: call to `ets:tab2list` instead of `ets:lookup`.
- Check a different table, replace `sbgFilterTable` by `smmFilter`.

- Don't generalise: too many parameters, how to name?

```
check_filter_exists_in_sbgFilterTable(FilterKey, FilterName, FilterState) ->
  ?OM_CHECK([#sbgFilterTable{key=FilterKey,
                 sbgFilterName=FilterName,
                 sbgFilterState=FilterState}],
     ?MP_BS, ets, lookup, [sbgFilterTable, FilterKey]).
```

Step 6

Symbolic calls to deprecated code: `erlang:module_loaded`

```
erlang:module_loaded(M) -> true | false
code:is_loaded(M) -> {file, Loaded} | false
```

Re-define the function `code_is_loaded`:

```
code_is_loaded(BS, ModuleName, false) ->
 ?OM_CHECK(false, BS, code, is_loaded, [ModuleName]).
code_is_loaded(BS, ModuleName, true) ->
 ?OM_CHECK({file, atom_to_list(ModuleName)}, BS, code,
                  is_loaded, [ModuleName]).
```

Step 7

Different checks: `?OM_CHECK` vs `?CH_CHECK`

```
code_is_loaded(BS, om, ModuleName, false) ->
 ?OM_CHECK(false, BS, code, is_loaded, [ModuleName]).
code_is_loaded(BS, om, ModuleName, true) ->
 ?OM_CHECK({file, atom_to_list(ModuleName)}, BS, code,
                  is_loaded, [ModuleName]).
```

But the calls to `?OM_CHECK` have disappeared at step 6 …

… a case of premature generalisation!

Need to inline `code_is_loaded/3` to be able to use this …

Step 10

‘Widows’ and ‘orphans’ in clone identification.

```
new_fun(FilterName, NewVar_1) ->
  FilterKey = ?SMM_CREATE_FILTER_CHECK(FilterName),
  %%Add rulests to filter
  RuleSetNameA = "a",
  RuleSetNameB = "b",
  RuleSetNameC = "c",
  RuleSetNameD = "d",
  ... 16 lines which handle the rules sets are elided ...
  %%Remove rulesets
  NewVar_1,
{RuleSetNameA, RuleSetNameB, RuleSetNameC, RuleSetNameD, FilterKey}.
```

Avoid passing commands as parameters?

```
new_fun(FilterName, FilterKey) ->
  %%Add rulests to filter
  RuleSetNameA = "a",
  RuleSetNameB = "b",
  RuleSetNameC = "c",
  RuleSetNameD = "d",
  ... 16 lines which handle the rules sets are elided ...
  %%Remove rulesets

{RuleSetNameA, RuleSetNameB, RuleSetNameC, RuleSetNameD}.
```

Also at step 11.

Steps 14+

Similar code detection (default params):

16 clones, each duplicated once.

193 lines in total: get 145 line reduction.

Reduce similarity to 0.5 rather than the default of 0.8: 47 clones.

Other refactorings: data etc.

Going further

Property extraction

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

Support property extraction from 'free' and EUnit tests.

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

Identifying state machines implicit in sets of test cases.

Refactoring and testing

Refactor tests themselves, e.g.

- Turn 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 into EUnit fixtures.

Respect test code in EUnit, QuickCheck and Common Test …

… and refactor tests along with code refactoring.

Conclusions

Possible to improve code using clone removal techniques …

… but only with expert involvement.

Not just test code … but it's particularly applicable there.

Hands on demo and tutorial tomorrow.

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