The pragmatics of clone detection and elimination

Simon Thompson, Huiqing Li
Andreas Schumacher

University of Kent,UK and Ericsson AB

ProTest
property based testing

University of Kent | Computing

A story about …

A story about …

… a tool

A story about …

… a tool

… a concept

A story about …

… a tool

… a concept

… and practice

A story about …

… a tool: Wrangler, for refactoring Erlang

… a concept

… and practice

A story about …

… a tool: Wrangler, for refactoring Erlang

… a concept: code clones

… and practice

A story about …

… a tool: Wrangler, for refactoring Erlang

… a concept: code clones

… and practice: case studies with Ericsson

Insights about …

Insights about …

… how to design (refactoring) tools

Insights about …

… how to design (refactoring) tools

… what “code clone” might mean

Insights about …

… how to design (refactoring) tools

… what “code clone” might mean

… practice of clone detection and elimination

Erlang / refactoring / Wrangler

Erlang

Functional language.

Concurrency built-in.

OTP for fault-tolerance and robustness.

Dynamic language: hot code loading, …

Good tool ecosystem.

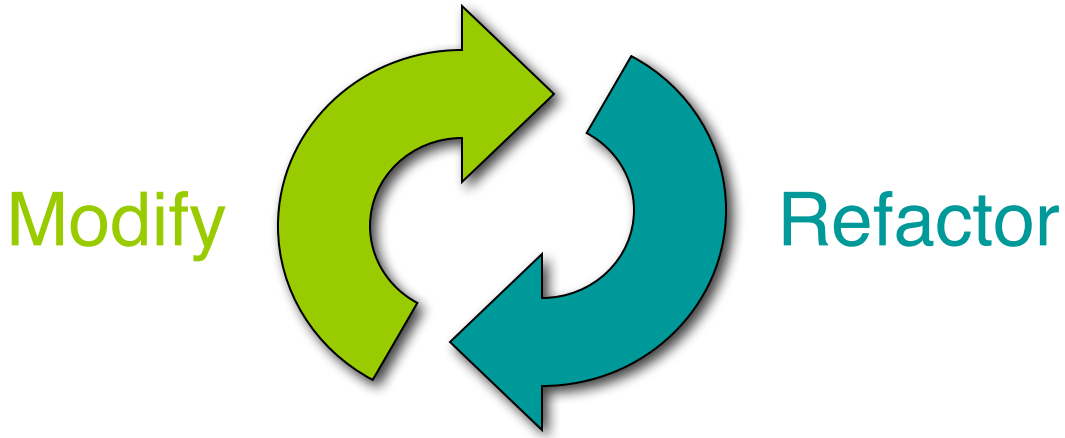
Open source.

Industrial take-up: WhatsApp … SMEs.

Ericsson support.

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

Wrangler refactoring tool

Structural, process, macro refactorings.

Integrated into Emacs, Eclipse, …

Multiple modules.

Testing-aware.

Refactoring = Condition + Transformation

Implement the simple …
… report the complex.

Make it extensible!

Usability?

Clone detection

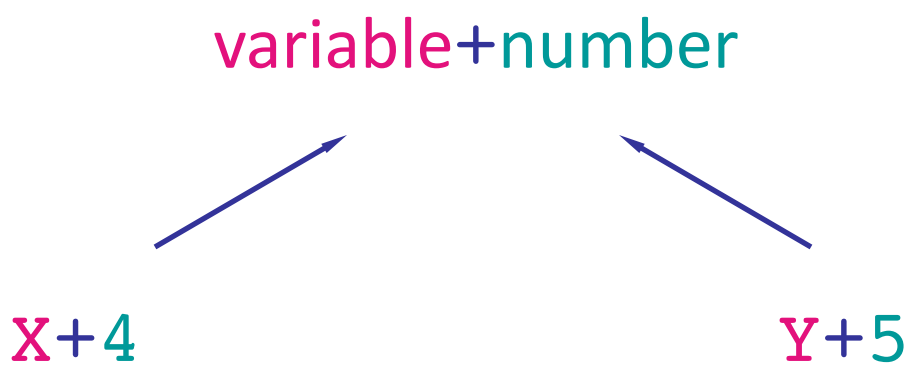
Duplicate code considered harmful

It's a bad smell …

- increases chance of bug propagation,
- increases size of the code,
- increases compile time, and,
- increases the cost of maintenance.

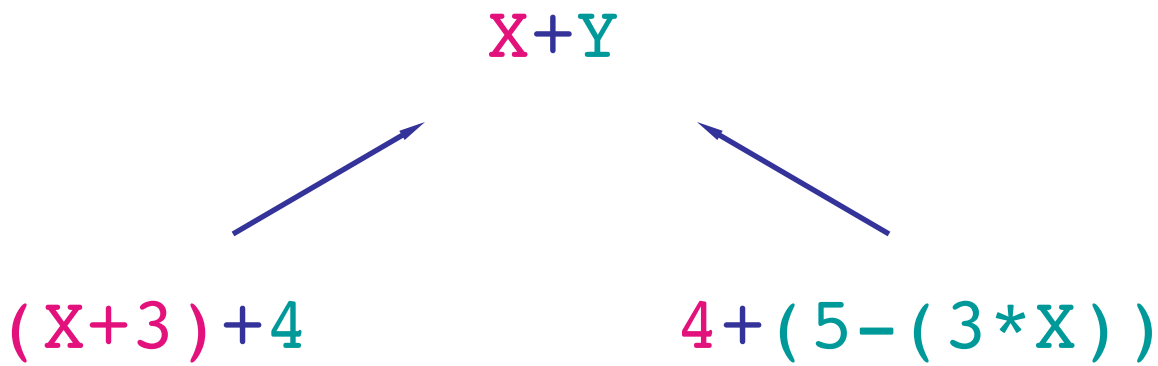
But … it's not always a problem.

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.

Example: clone candidate

```
S1 = "This",        S1 = "This",          D1 = [1],     D1 = [X+1],
S2 = " is a ",      S2 = "is another ",   D2 = [2],     D2 = [5],
S3 = "string",      S3 = "String",        D3 = [3],     D3 = [6],
[S1,S2,S3]          [S3,S2,S1]            [D1,D2,D3]    [D3,D2,D1]
```

```
? = ?,
? = ?,
? = ?,
[?,?,?]
```

Example: clone from sub-sequence

```
S1 = "This",        S1 = "This",          D1 = [1],     D1 = [X+1],
S2 = " is a ",      S2 = "is another ",   D2 = [2],     D2 = [5],
S3 = "string",      S3 = "String",        D3 = [3],     D3 = [6],
[S1,S2,S3]          [S3,S2,S1]            [D1,D2,D3]    [D3,D2,D1]
```

```
new_fun(NewVar_1,
        NewVar_2,
        NewVar_3) ->
  S1 = NewVar_1,
  S2 = NewVar_2,
  S3 = NewVar_3,
  {S1,S2,S3}.
```

Example: sub-clones

```
S1 = "This",        S1 = "This",          D1 = [1],     D1 = [X+1],
S2 = " is a ",      S2 = "is another ",   D2 = [2],     D2 = [5],
S3 = "string",      S3 = "String",        D3 = [3],     D3 = [6],
[S1,S2,S3]          [S3,S2,S1]            [D1,D2,D3]    [D3,D2,D1]
```

```
new_fun(NewVar_1,
        NewVar_2,
        NewVar_3) ->
  S1 = NewVar_1,
  S2 = NewVar_2,
  S3 = NewVar_3,
  [S1,S2,S3].
```

```
new_fun(NewVar_1,
        NewVar_2,
        NewVar_3) ->
  S1 = NewVar_1,
  S2 = NewVar_2,
  S3 = NewVar_3,
  [S3,S2,S1].
```

What makes a clone?

- Thresholds
- Threshold values and defaults

Thresholds

- Number of expressions

Thresholds

- Number of expressions
- Number of tokens

Thresholds

- Number of expressions
- Number of tokens
- Number of variables introduced

Thresholds

- Number of expressions
- Number of tokens
- Number of variables introduced
- Similarity = $\min_{i=1..n}(\text{size}(AU)/\text{size}(E_i))$

Threshold values

- Number of expressions ≥ 5
- Number of tokens ≥ 20
- Number of variables introduced ≤ 4
- Similarity = $\min_{i=1..n}(\text{size}(AU)/\text{size}(E_i)) \geq 0.8$

What makes a clone?

Which thresholds and what threshold values?

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.

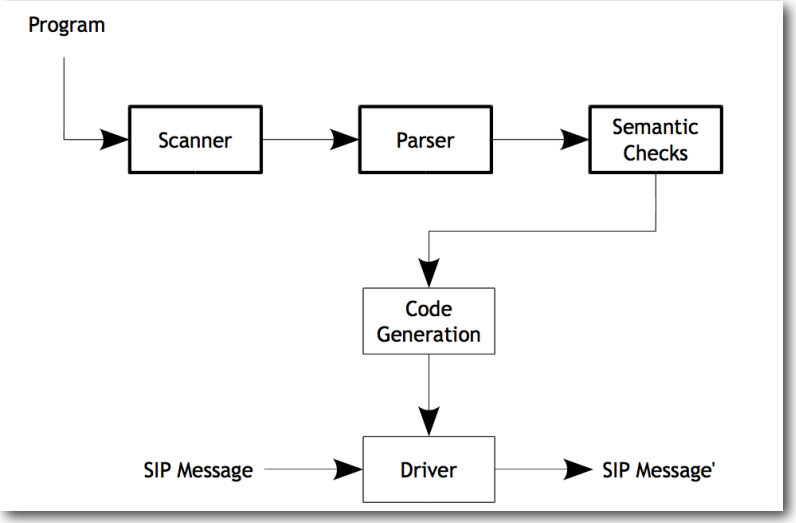
The SIP Case Study

SIP case study

ERICSSON

Session Initiation Protocol

SIP message manipulation allows rewriting rules to transform messages.



`smm_SUITE.erl`

2658 LOC.

ProTest property based testing

University of Kent Computing

Why test code particularly?

Many people touch the code.

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

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

“Who you gonna call?”

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

… what results is of no value at all.

Need to call in the domain experts.

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		

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)

A var by any other name …

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),
```

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)

Bottom up, not top down

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?

May choose a sub-clone

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

Use search mode to explore the nature of the sub-clone.

```
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

Avoid over-generalisation …

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]).
```

… but consolidate

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 …

‘Widows’ and ‘orphans’

Lines of code “accidentally“ coincides with the real clone.

```
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}.
```

Refactoring ⇒ comprehension

The process of naming *is* dependent on understanding the code …

… and that understanding can lead to some manual refactoring and so to larger clones being found (8.1.4).

Also identifies bugs: ‘recovery’ / ‘rovery’.

And for the refactoring tool …

Look across modules.

Improve the reports (parameter values).

Parameter order.

Add some refactorings: e.g. inlining.

And for the refactoring tool …

Look across modules.

Improve the reports (parameter values).

Parameter order.

Add some refactorings: e.g. inlining.

And make it incremental … workflow

And for the refactoring tool …

Look across modules.

Improve the reports (parameter values).

Parameter order.

Add some refactorings: e.g. inlining.

And make it incremental … workflow

DSL for “scripting”

In the DSL

Transaction control

rename the function

rename the variables

replace all the instances

```
/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:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:1803.4-1809.71:
/Users/simonthompson/Desktop/StockholmAug09/code/smm_SUITE.erl:20
/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)
```

Tool + human

Clone detection and elimination needs tooling to make it practical …

Tool + human

Clone detection and elimination needs tooling to make it practical …

… but there has to be a human in the loop, irrespective of language, tool and application area.

Tool + human

The right notion of clone for a particular project comes from a complex space of parameters and thresholds.

Tool + human

The right notion of clone for a particular project comes from a complex space of parameters and thresholds.

Refactoring in practice relies on a set of complex choices and tradeoffs, which just can’t be automated.

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