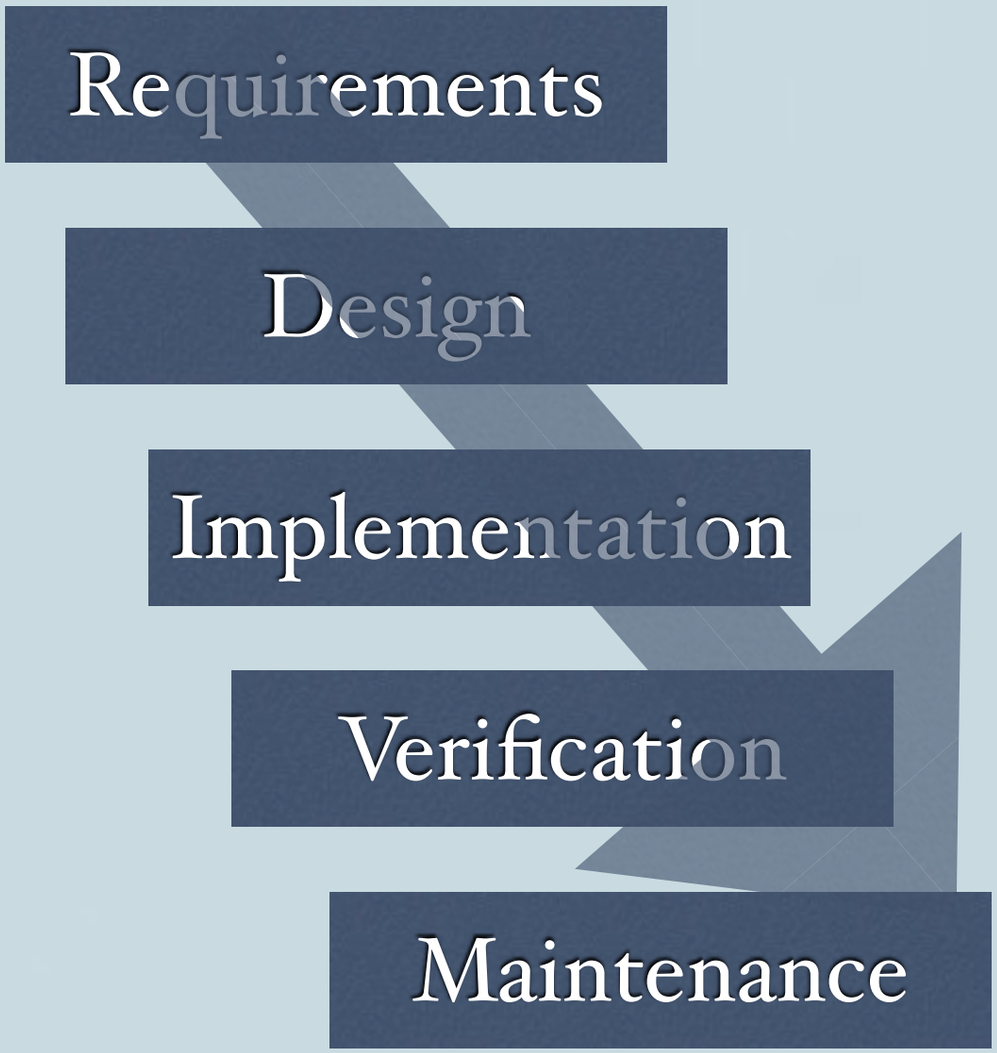
OCaml Companion Tools

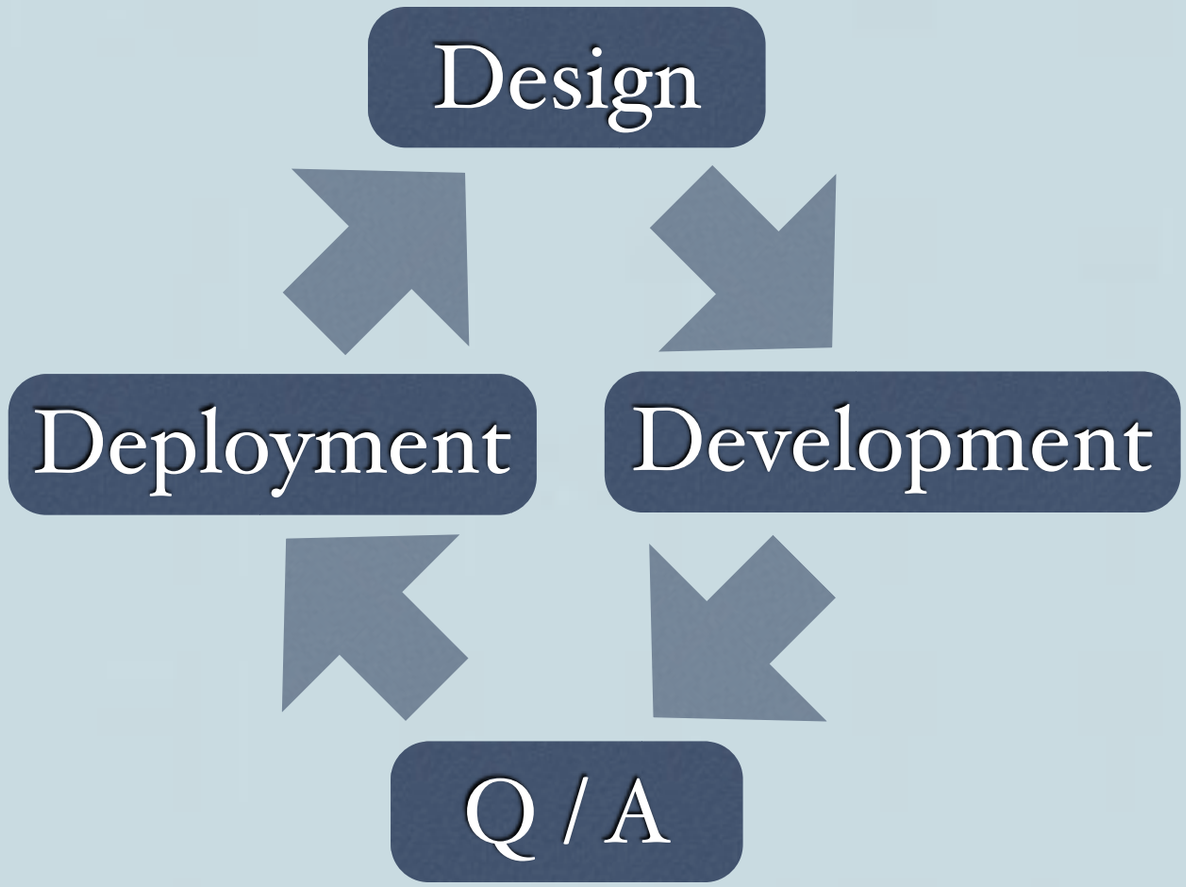
OCaml Users & Developers 2012

Copenhagen - 14 september 2012

Xavier Clerc - forum@x9c.fr

Whatever your process is...





Anyway, you will need to...

- Browse APIs
- Write specifications and/or tests
- Understand program behaviour
- Ensure that test is thorough
- Ensure that code is readable

Argot

- Extended `ocamldoc` generator
- Adds tags and style
- Provides search facilities
  - name-based
  - type-based
  - fulltext-based

of_int search

by name
by regexp
by type by type, using manifest
by full text

```
Int32.of_int: int -> int32
```
Convert the given integer (type int) to a 32-bit integer (type int32).

```
Int64.of_int: int -> int64
```
Convert the given integer (type int) to a 64-bit integer (type int64).

```
Nativeint.of_int: int -> nativeint
```

convert integer big search

by name
by regexp
by type by type, using manifest
by full text

```
Big_int.big_int_of_int: int -> Big_int.big_int
```
Convert a small integer to a big integer.

```
Big_int.big_int_of_int32: int32 ->
Big_int.big_int
```
Convert a 32-bit integer to a big integer.

```
val big_int_of_
```
Convert a sma

```
val is_int_big_
```
Test whether t
integer (type i
returns true i
is_int_big_

```
val int_of_big_
```
Convert a big i
"int_of_bi

```
val big_int_of_
```
Convert a 32-b

of_int.* search

by name
by regexp
by type by type, using manifest
by full text

```
Int32.of_int: int -> int32
```
Convert the given integer (type int) to a 32-bit integer (type int32).

```
Int64.of_int: int -> int64
```
Convert the given integer (type int) to a 64-bit integer (type int64).

```
Int64.of_int32: int32 -> int64
```
Convert the given 32-bit integer (type int32) to a 64-bit integer (type int64).

```
Nativeint.of_int: int -> nativeint
```
Convert the given integer (type int) to a native integer (type nativeint).

```
Nativeint.of_int32: int32 -> nativeint
```
Convert the given 32-bit integer (type int32) to a native integer.

float -> float -> float search

by name
by regexp
by type by type, using manifest
by full text

```
Pervasives.( ** ): float -> float -> float
```
Exponentiation.

```
Pervasives.( *. ): float -> float -> float
```
Floating-point multiplication

```
Pervasives.(+.): float -> float -> float
```
Floating-point addition

```
Pervasives.(-.): float -> float -> float
```
Floating-point subtraction

```
Pervasives.(/.): float -> float -> float
```
Floating-point division.

```
Pervasives.atan2: float -> float -> float
```
atan2 y x returns the arc tangent of y / x.

```
Pervasives.copysign: float -> float -> float
```
copysign x y returns a float whose absolute value is that of x and whose sign is that of y.

by type by type, using manifest
by full text

```
Complex.arg: Complex.t -> float
```
Argument

```
Complex.norm: Complex.t -> float
```
Norm: sqrt(x * x + y * y).

```
Complex.norm2: Complex.t -> float
```
Norm squared: given x = a + i * b, returns a * a + b * b.

```
Pervasives.( ** ): float -> float -> float
```
Exponentiation.

```
Pervasives.( *. ): float -> float -> float
```
Floating-point multiplication

```
Pervasives.(+.): float -> float -> float
```
Floating-point addition

```
Pervasives.(-.): float -> float -> float
```
Floating-point subtraction

```
Pervasives.(/.): float -> float -> float
```
Floating-point division.

```
Pervasives.atan2: float -> float -> float
```
atan2 y x returns the arc tangent of y / x.

Kaputt

- Combinator-based test library
- Tests can be stored in `.mlt` files
- Provides support for various kinds of tests
  - assertion (xUnit-like)
  - enumeration (SmallCheck-like)
  - specification (QuickCheck-like)

Kaputt

```
        OCaml version 4.00.0

# #directory "+kaputt";;
# #load "kaputt.cma";;
# open Kaputt.Abbreviations;;
# let gen_list = Gen.list (Gen.make_int 0 10) Gen.int;;
val gen_list : int list Kaputt.Generator.t = (<fun>, <fun>)
# let rec is_sorted = function
  | fst :: snd :: tl -> (fst <= snd) && is_sorted (snd :: tl)
  | _ -> true;;
val is_sorted : 'a list -> bool = <fun>
# check gen_list (List.sort compare) [Spec.always ==> is_sorted];;
Test 'untitled no 1' ... 100/100 cases passed
- : unit = ()
```

Kaputt

```
# let () =
    Test.add_simple_test
      (fun () ->
        let eq_int_list = Assert.make_equal_list (=) string_of_int in
        let f = Mock.from_function succ in
        let i = [0; 1; 2; 0] in
        let o = List.map (Mock.func f) i in
        let o' = [1; 2; 3; 1] in
        eq_int_list o' o;
        eq_int_list i (Mock.calls f);
        Assert.equal_int 4 (Mock.total f));;
```

Bolt

- both `camlp4`- and API-based log library
- Allows to configure *how/where* to log at runtime
- Layouts for various formats
  - from bare text to XML
  - Pajé (for multi-threads)
  - Daikon (for invariant computation)

```
let cat filename =
  LOG "printing file %S" filename LEVEL TRACE;
  let channel = open_in filename in
  try
    while true do
      let line = input_line channel in
      print_endline line
    done
  with _ -> ()
```

```
let () =
  LOG "application start" LEVEL TRACE;
  let len = Array.length Sys.argv in
  for i = 1 to pred len do
    Aux.cat Sys.argv.(i)
  done;
  LOG "application end" LEVEL TRACE
```

```
logger "" {
  level = trace;
  filter = all;
  layout = simple;
  output = file;
  name = "log";
}

logger "" {
  level = fatal;
  filter = all;
  layout = html;
  output = file;
  name = "log-important";
}

logger "Main" {
  level = trace;
  filter = all;
  layout = default;
  output = file;
  name = "log-main";
}

logger "Aux" {
  level = trace;
  filter = all;
  layout = default;
  output = file;
  name = "log-aux";
}
```

Welcome | Drag & Drop XML log files here | Zeroconf | chainsaw-log | file-log4j.xml

Refine focus on:

Root Logger

ID	Timestamp	Level	Logger	Message
1	2012-08-13 21:04:4...			application start
2	2012-08-13 21:04:4...			func0(3)
3	2012-08-13 21:04:4...			func0(7)
4	2012-08-13 21:04:4...			application end

Level	TRACE
Logger	
Time	2012-08-13 21:04:42,990
Thread	0
Message	application end
NDC	null
Class	Unknown
Method	unknown()
Line	11
File	source.ml
Properties	{{hostname,file}{application,log4j.xml}{log4jid,4}}
Throwable	

Receiver
No Rece

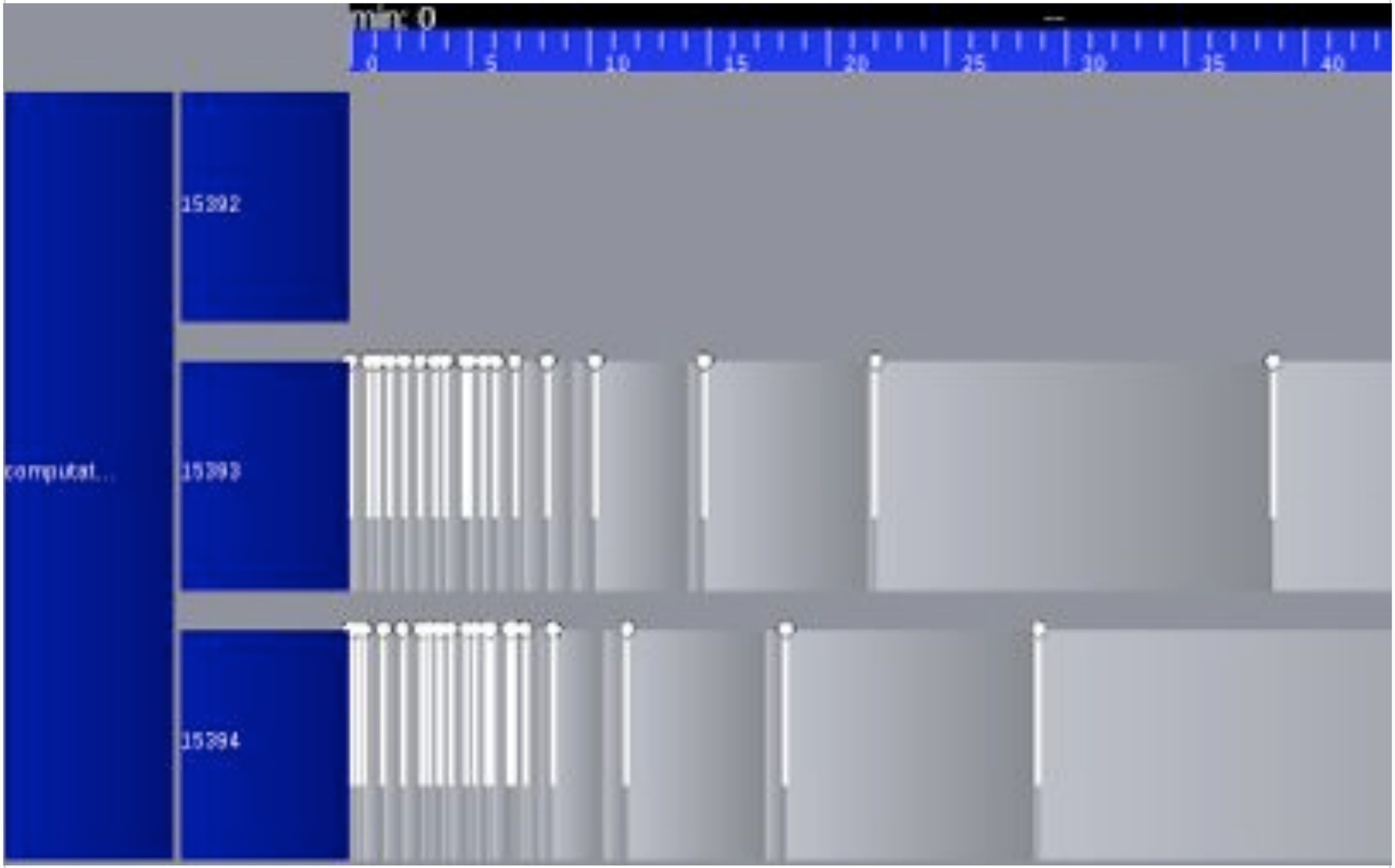
Property

0 hidden loggers

Looks like ZeroConf stuff is available... WooHoo!

0

0:0


min: 0
0
5
10
15
20
25
30
35
40
15392
15393
15394
compiutat...

Bisect

- `camlp4`-based code instrumenter
- Allows to tell whether an expression has been evaluated
- Reports in various formats
  - bare text (statistics)
  - HTML (code replica)
  - XML (EMMA-compatible)

Statistics:

kind	coverage	kind	coverage
binding	2 / 2 (100%)	class expression	0 / 0 (-%)
sequence	2 / 2 (100%)	class initializer	0 / 0 (-%)
for	0 / 0 (-%)	class method	0 / 0 (-%)
if/then	0 / 0 (-%)	class value	0 / 0 (-%)
try	1 / 1 (100%)	toplevel expression	0 / 0 (-%)
while	1 / 1 (100%)	lazy operator	0 / 0 (-%)
match/function	1 / 1 (100%)		

Source:

fold all unfold all

```
!000001| let cat filename =
!000002|   (*[1]*)LOG 'printing file %S' filename LEVEL TRACE;
!000003|   (*[1]*)let channel = open_in filename in
!000004|   (*[1]*)try
!000005|     (*[1]*)while true do
!000006|       (*[10]*)let line = input_line channel in
!000007|       (*[9]*)print_endline line
!000008|     done
!000009|   with _ -> (*[1]*)()
```

Legend:

none code - line containing no point
none code - line containing only visited points
none code - line containing only unvisited points
none code - line containing both visited and unvisited points

Mascot

- Customizable style-checker (configuration, plugins)
- Complementary to compiler warnings
- Wide range of checks
  - typography, documentation
  - code smells, interface smells
  - metrics, etc.

```
category typography {
  line_length = { maximum = 50; };
  tab_character = true;
  trailing_white_space = true;
}

category code {
  empty_for = true;
  ignore_unit = true;
  no_effect_assignment = true;
}
```

```
let iter : ('a -> unit) -> 'a array
  for i = 0 to Array.length a - 1 do
    ()
  done

let print : ('a -> string) -> 'a arr
  let i = ref 0 in
  while !i < Array.length a do
    print_endline (f a.(i));
    i := i
  done
```

Summary: 0 warning, 6 errors, and 0 info

src/style.ml

typography	line_length	line 1	colum
code	empty_for	line 2	colum
typography	tab_character	line 3	colum
typography	line_length	line 6	colum
typography	trailing_white_space	line 9	colum
code	no_effect_assignment	line 10	colum

```
line 1, column 56, error: line is too long (56 instead of 50)
line 2, column 2, error: empty 'for' loop
line 3, column 0, error: tab character
line 6, column 59, error: line is too long (59 instead of 50)
line 9, column 29, error: trailing white space
line 10, column 4, error: assignment with no effect
```

Using ocamlbuild

```
open Ocamlbuild_plugin
open Ocamlbuild_pack

let rec copy_mlt_files path =
  let elements = Pathname.readdir path in
  Array.iter
    (fun p ->
      if Pathname.is_directory (path / p) then
        copy_mlt_files (path / p)
      else if Pathname.check_extension p "mlt" then
        let src = path / p in
        let dst = !Options.build_dir / path / p in
        Shell.mkdir_p (!Options.build_dir / path);
        Pathname.copy src dst
      else
        ())
    elements

let getenv name default =
  String.uppercase
    (try
      Sys.getenv name
    with _ -> default)

let lib_dir = "/path/to/lib/ocaml"
```

```
let test_code = getenv "TEST" "OFF"
let log_level = getenv "LOG_LEVEL" "NONE"
let coverage = getenv "COVERAGE" "OFF"

let () =
  dispatch begin function
    | After_rules ->
      copy_mlt_files "src";
      flag ["ocaml_tools"; "pp"]
        (S [A"kaputt_pp.byte";
            (match test_code with "ON" -> A"on" | _ -> A"off");
            A"camlp4o"; A"str.cma";
            (match coverage with
            | "ON" -> A(lib_dir ^ "/bisect/bisect_pp.cmo")
            | _ -> N);
            A(lib_dir ^ "/bolt/bolt_pp.cmo");
            A"-level"; (A log_level) ]);
      flag ["ocaml_tools"; "compile"]
        (S [A"-I"; A(lib_dir ^ "/bolt");
            A"-I"; A(lib_dir ^ "/bisect");
            A"-I"; A(lib_dir ^ "/kaputt")]);
      flag ["ocaml_tools"; "link"; "byte"]
        (S [A"-I"; A(lib_dir ^ "/bolt"); A"bolt.cma";
            A"-I"; A(lib_dir ^ "/bisect"); A"bisect.cma";
            A"-I"; A(lib_dir ^ "/kaputt"); A"kaputt.cma"]);
    | _ -> ()
end
```

Using `ocamlbuild`

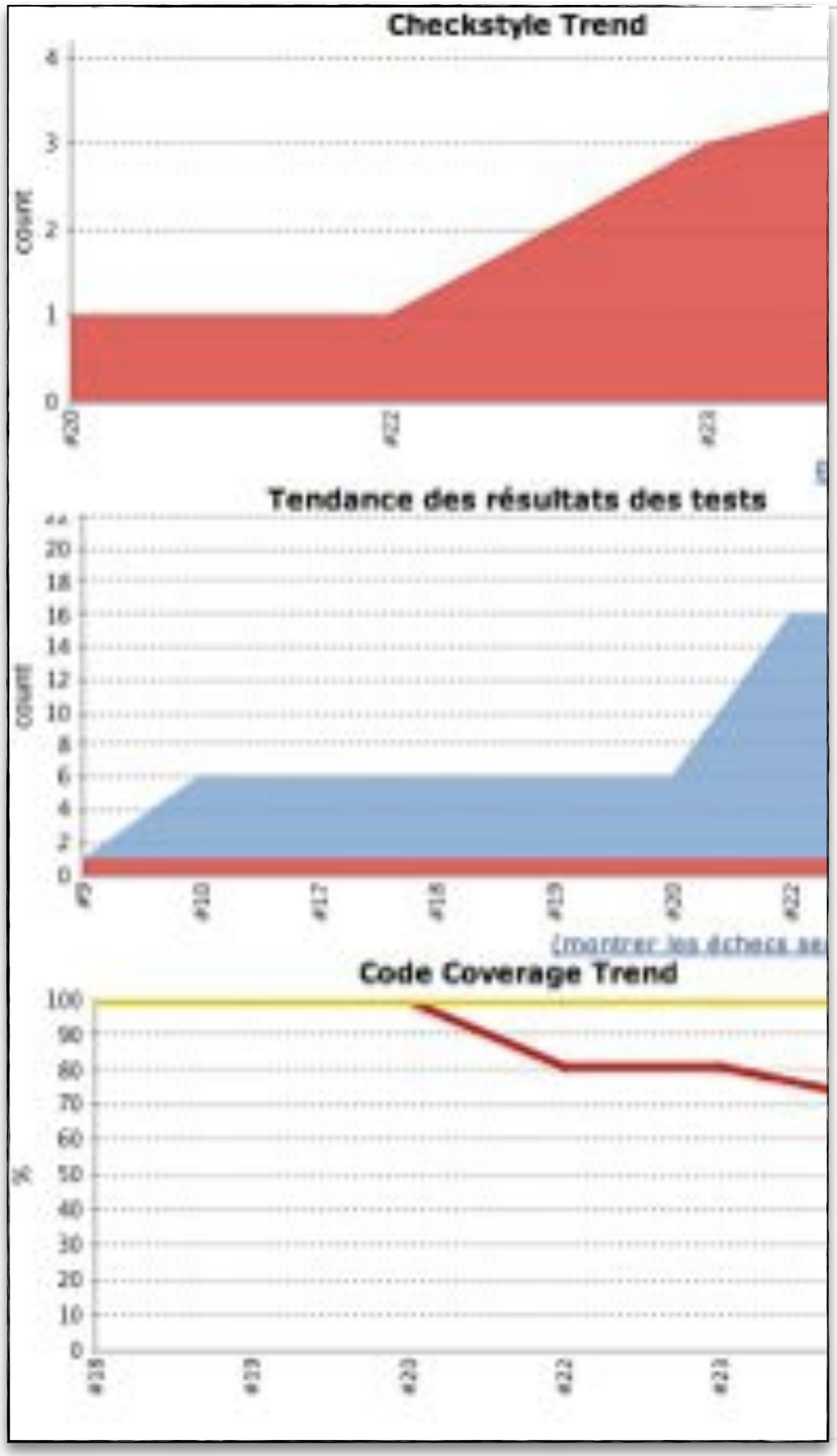
- Tag files with « `ocaml_tools` »
- Use environment variables to select features

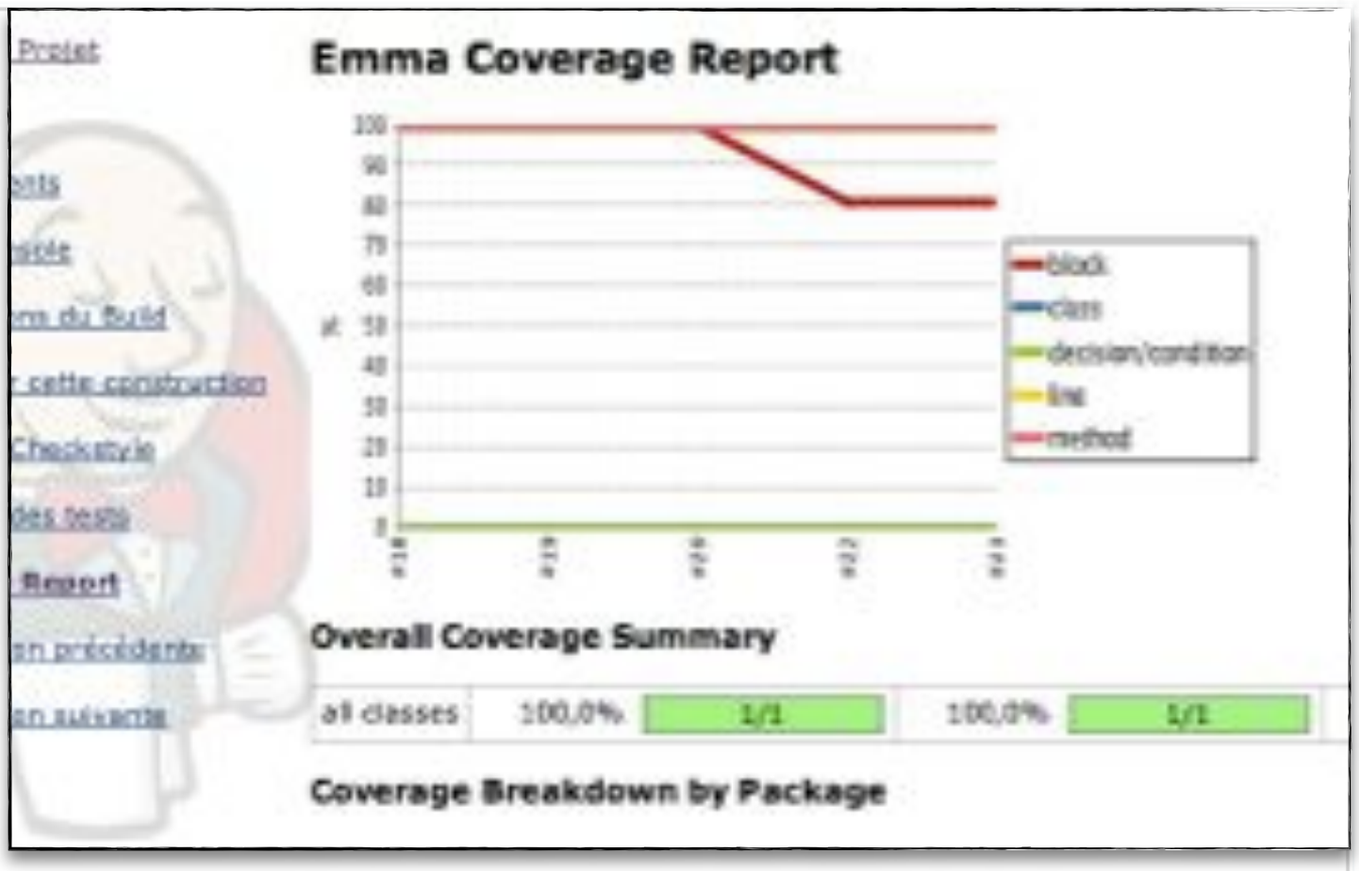
- `TEST=on LOG_LEVEL=trace COVERAGE=on ocamlbuild (...)`
- `BOLT_CONFIG=config BISECT_FILE=coverage prog`

Using Jenkins

- Bisect has EMMA-compatible output
- Kaputt has jUnit-compatible output
- Mascot has CheckStyle-compatible output

- Easy to plug into Jenkins
- Jenkins will take care of project history


Checkstyle Trend
count
#20
#21
#23
Tendance des résultats des tests
count
#9
#10
#17
#18
#19
#20
#22
(montrer les échecs se
Code Coverage Trend
%
#28
#29
#23
#22
#26


Projet
Emma Coverage Report
#
block
class
decision/condition
line
method
Overall Coverage Summary
all classes
100,0%
1/1
100,0%
1/1
Coverage Breakdown by Package


Projet
Résultats des tests
1 échecs (±0)
Tous les tests qui ont échoué
Nom du test
>>> Kaputt Report.random test
Tous les tests
Package
(root)

Thanks!

Questions?