klarna

distel
the first ten years

mats cronqvist
masse@klarna.com

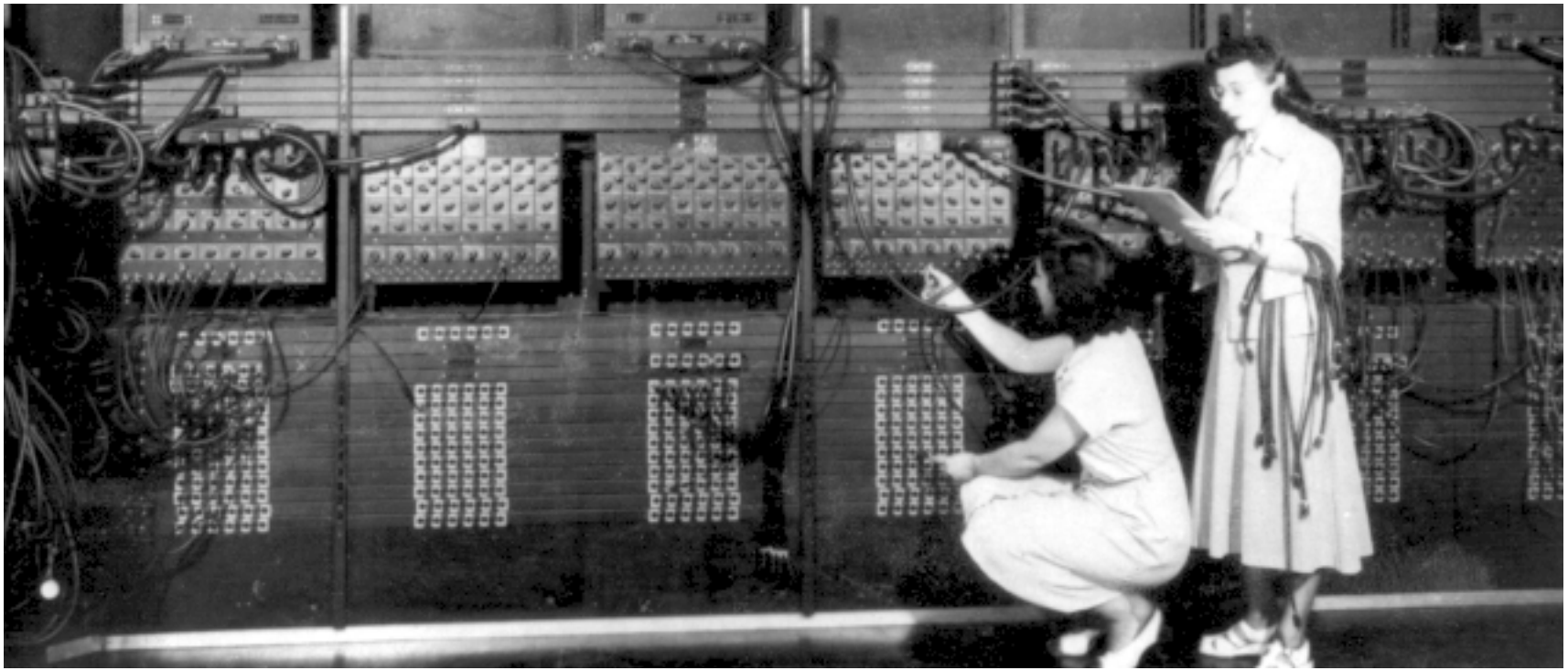
this talk

klarna

- editing code
- emacs
- erlang in emacs
- distel

how we write code - 40's

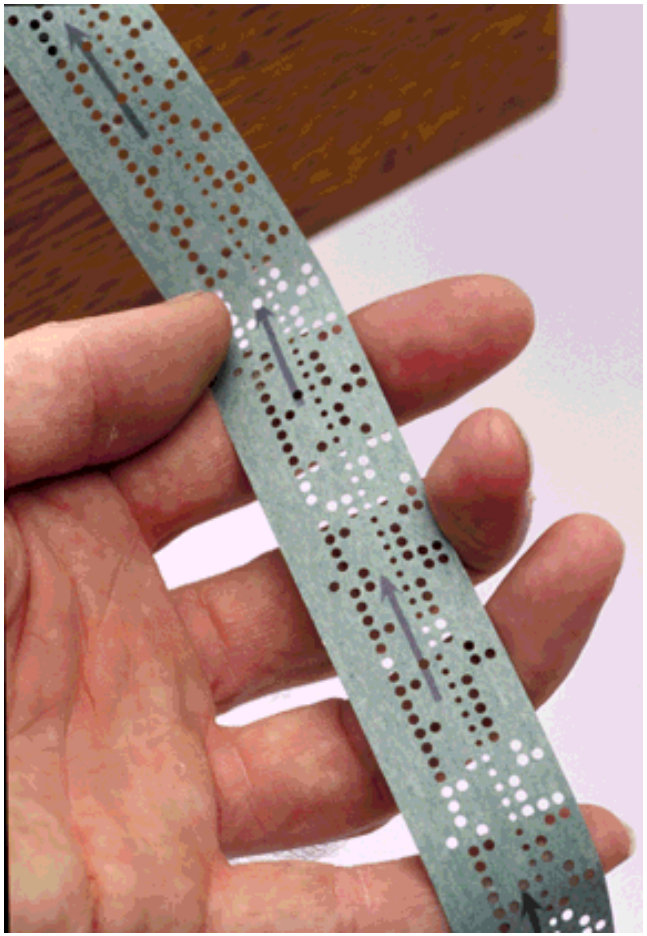
klarna



ENIAC programmers, 1945 (U.S. Army)

how we write code - 50s

klarna



how we write code - 60s

klarna

interactive line editors.

- TECO (1963)
- QED (1965)

*SHello$0TT$$ Search for "Hello" and print the line printf("Hello world!\n"); The line *-5DIGoodbye$0TT$$ Delete "Hello", insert "Goodbye", and print the line printf("Goodbye world!\n"); The updated line

how we write code -70s

klarna

Visual editors

- vi (1976)
- emacs (1976)

how we write code - 90s

klarna

Visual Studio (1995)

- code editor supporting IntelliSense
- code refactoring.
- integrated debugger works both as a source-level debugger and a machine-level debugger.
- forms designer for building GUIapplications,
- web designer,
- class designer
- database schema designer
- source-control systems (like Subversion and Visual SourceSafe)

how we write code

klarna

by editing test in files.
nothing's changed since card punches disappeared.

Stevey on IDEs

klarna

People in the industry are very excited about[ideas]such as IDEs that can manipulate code as "algebraic structures", and search indexes, and so on.
These people tend to view code bases much the way construction workers view dirt: they want great big machines that can move the dirt this way and that.

http://steve-yegge.blogspot.se/2007/12/codes-worst-enemy.html

future?

klarna

Not much has happened since mid-80s.

- Inventing on Principle - Bret Victor
- Light Table - Chris Granger

"Light Table is based on a very simple idea: we need a real work surface to code on, not just an editor and a project explorer. We need to be able to move things around, keep clutter down, and bring information to the foreground in the places we need it most."

[http://vimeo.com/36579366]
[http://www.chris-granger.com/2012/05/21/the-future-is-specific]

editing erlang

klarna

- eclipse
- vim
- emacs
  - the preferred tool of the original Erlang crew
  - has a comprehensive erlang mode
  - defines the canonical indentation

erlang in emacs

klarna

- erlang mode (Anders Lindgren, ca. 1995)
  - syntax highlighting
  - navigation
  - indentation
  - compilation
  - inferior shell
  - ...

trivial emacs extensibility

emacs is extensible

```
(if (file-exists-p "Makefile")
    "make -k"
  (concat
 "erlc "
    (if (file-exists-p "../ebin") "-o ../ebin " "")
    (if (file-exists-p "../inc") "-I ../inc " "")
    "+debug_info -W "
    buffer-file-name))
```

non-trivial emacs extensibility

E.g., it is possible to implement the Erlang Distribution on top of it.

Thats's what this fine looking feller did back in 2002.
Luke Gorrie.

highly ambitious

klarna

"Distel is an Emacs-based user-interface toolkit for Erlang. We introduce “Emacs nodes” using the Erlang inter-node distribution protocol, and make communication natural by extending Emacs Lisp with Erlang’s concurrent programming model."

"[The] features we selected are processes, pattern matching, and distribution, and they are reproduced faithfully at a high level, though many details differ."

distel example

```
(defun spawn-counter ()
  (erl-spawn
    (erl-register 'counter)
    (&counter-loop 1)))

(defun &counter-loop (count)
  (erl-receive (count)
    ((msg (message "Got msg #%S: %S"
           count msg)))
    (&counter-loop (+ count 1))))
```

continuation-passing

"Most importantly, erl-receive never returns. Instead it bundles up the execution state and throw's it directly back up to a scheduler loop, bypassing any code on the stack."

"Because erl-receive doesn't return, and nor do functions that call it, they should only be tail-called – called as the last thing a function does."

caveats

Distel is a bit old. And tends to get attention only in the areas that are used by highly motivated hackers.

- Has problems on MacOS.
- Probably doesn't work at all on Windows.
- Some bits are virtually never used, and rotted.

...but some bits work very well.

handy

`erl-reload-module (C-d C-d L)`

Reload an Erlang module, given by name in the minibuffer.

`erl-reload-modules (C-c C-d r)`

Reload all modules that are out of date.

`erl-find-doc (C-c C-d z)`

Show the signature of the function under point.

`file:read_file_info(Filename) -> {ok, FileInfo} | {error, Reason}`

hither and yon

Dynamic “TAGS”

"Distel includes a small source code cross referencer for Erlang. The basic feature is to jump from a function call in a program to the definition of that function."

`erl-find-source-under-point (M-.)` `erl-find-source-unwind (M-,)`

scribbles

Interactive Sessions

"An Interactive Session buffer is to Erlang as the *scratch* buffer is to Emacs Lisp – a scratchpad where code snippets can be hacked and executed."

squash

Debugger

"An Erlang debugger interface, called edb, is also included with Distel. This uses the same interpreter-based back-end as the OTP debugger application, but replaces the Tk-based front-end with an Emacs interface."

outro

https://massemanet@github.com/massemanet/distel.git

distel-ie

```
foo(Cmd) ->
  string:tokens(os:cmd(Cmd),"\n").

[foo(C) || C<- ["ls","date"]].


 -:-> [["elisp","foo.beam","foo.erl",
        "rpmbuild","user_default.beam"
       ["Mon May 28 12:57:09 CEST 2012
```

distel-edb

```
os(Cmd) ->
  lists:foreach(fun(X)->wr("~s~n",X)end,stri

wr(E) -> wr("~p.~n",E).
=>(F,E) -> wr(user,F,E).
wr(FD,F,E) -> io:fwrite(FD,F,[E]).

redbug()->redbug:help().
redbug(A,B,C)->redbug:start(A,B,C).

bt(P) ->
  string:tokens(binary_to_list(e(2,process_i
-UU-:%*--F1  *edbproc <?.132.0> on foo@vagra
        E = "Mon May 28 13:17:09 CEST 2012"
        F = "~s~n"
```