CLIP
CLoning as
Industrial Practice

A research project involving UWaterloo and CA

```
const char *err = ap_check_cmd_context(cmd, GLOBAL_ONLY);
if (err != NULL) {
    return err;
}
ap_threads_per_child = atoi(arg);
if (ap_threads_per_child > thread_limit) {
    ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
                 "WARNING: ThreadsPerChild of %d exceeds ThreadLimit "
                 "value of %d", ap_threads_per_child,
                 thread_limit);
    ....
    ap_threads_per_child = thread_limit;
}
else if (ap_threads_per_child < 1) {
    ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
                 "WARNING: Require ThreadsPerChild > 0, setting to 1");
    ap_threads_per_child = 1;
}
return NULL;
```

```
const char *err = ap_check_cmd_context(cmd, GLOBAL_ONLY);
if (err != NULL) {
    return err;
}
ap_threads_per_child = atoi(arg);
if (ap_threads_per_child > thread_limit) {
    ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
                 "WARNING: ThreadsPerChild of %d exceeds ThreadLimit "
                 "value of %d threads,", ap_threads_per_child,
                 thread_limit);
    ....
    ap_threads_per_child = thread_limit;
}
else if (ap_threads_per_child < 1) {
          ap_log_error(APLOG_MARK, APLOG_STARTUP, 0, NULL,
                 "WARNING: Require ThreadsPerChild > 0, setting to 1");
          ap_threads_per_child = 1;
}
return NULL;
```

```
gnumeric_oct2bin (FunctionEvalInfo *ei, GnmValue const * const *argv) {
   return val_to_base (ei, argv[0], argv[1],
      8, 2,
      0, GNM_const(7777777777.0),
      V2B_STRINGS_MAXLEN | V2B_STRINGS_BLANK_ZERO);
}


gnumeric_hex2bin (FunctionEvalInfo *ei, GnmValue const * const *argv)
{
   return val_to_base (ei, argv[0], argv[1],
      16, 2,
      0, GNM_const(9999999999.0),
      V2B_STRINGS_MAXLEN | V2B_STRINGS_BLANK_ZERO);
}
```

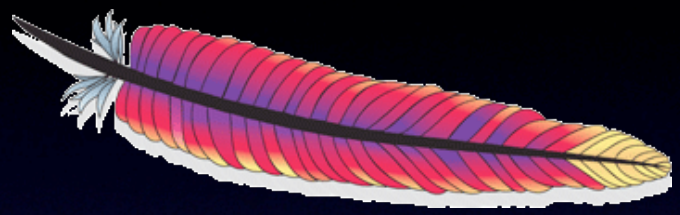
```
static PyObject *
py_new_RangeRef_object (const GnmRangeRef *range_ref){
   py_RangeRef_object *self;
   self = PyObject_NEW py_RangeRef_object,
     &py_RangeRef_object_type);
   if (self == NULL) {
      return NULL;
   }
   self->range_ref = *range_ref;
   return (PyObject *) self;
}
```

```
static PyObject *
py_new_Range_object (GnmRange const *range) {
   py_Range_object *self;
   self = PyObject_NEW (py_Range_object,
     &py_Range_object_type);
   if (self == NULL) {
      return NULL;
   }
   self->range = *range;
   return (PyObject *) self;
}
```

Why cloning is supposed to be bad

- It leads to code bloat + inconsistent maintenance
- It's a sign of inexperienced developers
  - And cruft accumulates as developers fear changing working code
- It's a sign of poor design / extensibility
  - Need to keep doing same kinds of things, but there's no easy way to automate it

... but what about ...



- Apache Portable Runtime (APR) subsystem
  - Portable impl of functionality that is typically platform dependent, such as file and network access
  - e.g., `fileio -> {netware, os2, unix, win32}`
  - Cloning is clearly obvious and is well documented!
    - Typical change: insertion of extra error checking or API calls.
  - Is this bad???

Cloning as an engineering tool

- Developers often use cloning!
  - If you understand the costs and risks, it can be used an engineering tool
  - ... but we need more study to better understand the phenomenon!

‘Cloning considered harmful’ … considered harmful [WCRE 06, EMSE 08]

1. Forking
   - Hardware variation
   - Platform variation
   - Experimental variation
2. Templating
   - Boilerplating
   - API / library protocols
   - Generalized programming idioms
   - Parameterized code
3. Customizing
   - Bug workarounds
   - Replicate + specialize

Forking: Platform variation

- Motivation:
  - Different platforms ⇒ very different low level details
  - Interleaving the platform-specific code in one place may be very complex
- Advantages of cloning:
  - Each (cloned) variant is simpler to maintain
  - No risk to stability of older variants
  - Platforms are likely to evolve independently, so maintenance is likely to be "mostly independent"

Forking: Platform variation

- Disadvantages of cloning:
  - Evolution in two dimensions: user requirements + platform support
  - Change to the interface level means changes to many files
- Management and long-term issues:
  - Factor out platform independent functionality as much as possible
  - Document the variation points and platform peculiarities
  - As number of platforms grows, the interface to the system hardens

Forking: Platform variation

- Structural manifestations:
  - Cloning usually happens at the file level.
    - Clones are often stored as files (or dirs) in the same source directory
- Well known examples:
  - Linux kernel "arch" subsystem
  - Apache Portable Runtime (APR)

Two case studies

Group	Pattern	Good	Harmful	Good	Harmful
Forking	Hardware variation	0	0	0	0
Forking	Platform variation	10	0	0	0
Forking	Experimental variation	4	0	0	0
Templating	Boiler-plating	5	0	6	7
Templating	API	0	0	0	9
Templating	Idioms	0	12	1	1
Templating	Parameterized code	5	12	10	34
Customizing	Replicate + specialize	12	4	15	16
Customizing	Bug workarounds	0	0	0	0
Total		**36**	**28**	**32**	**67**

Apache httpd 2.2.4 - 60 Tokens

Gnumeric 1.6.3 - 60 Tokens

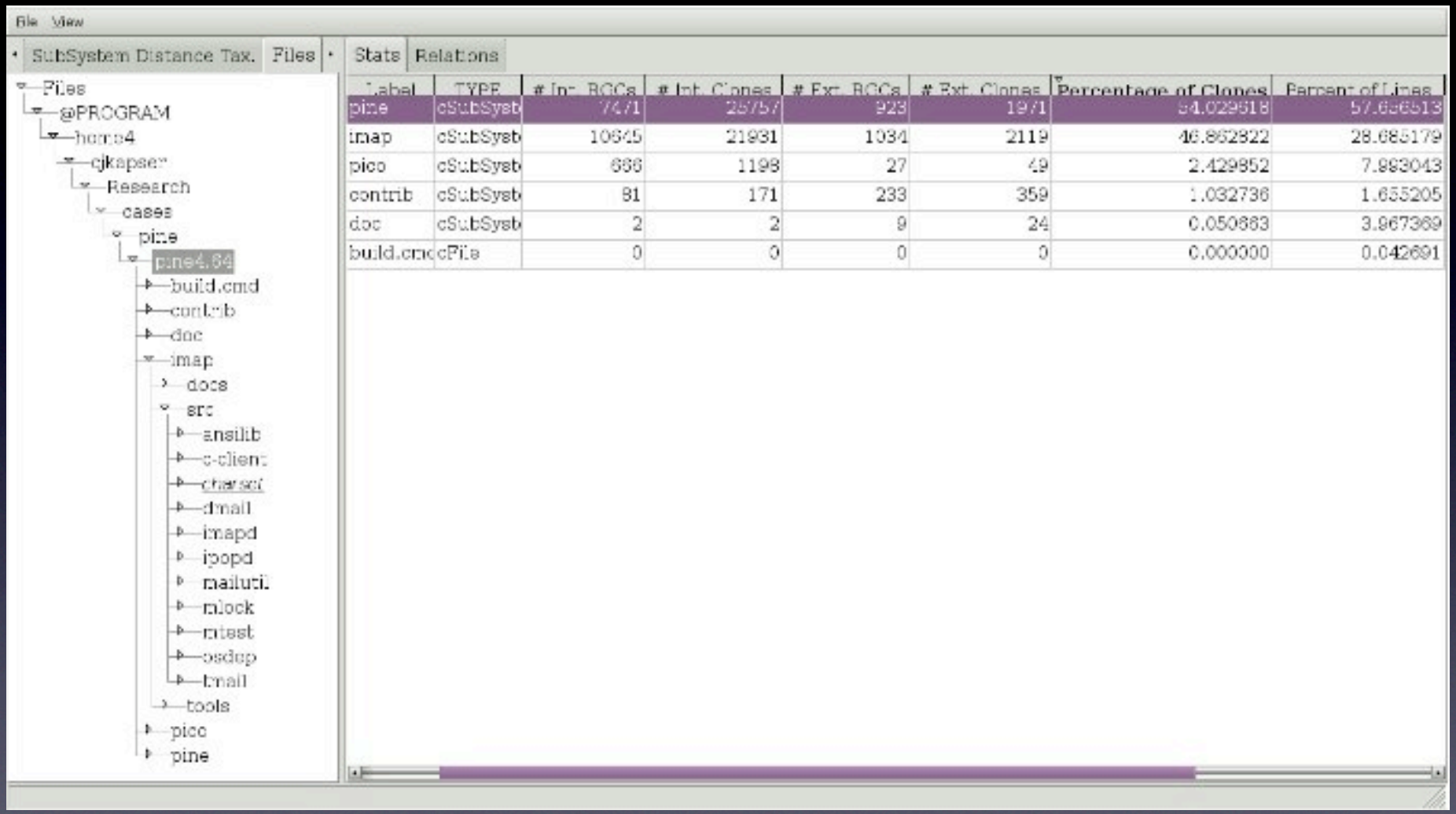
Research on code cloning

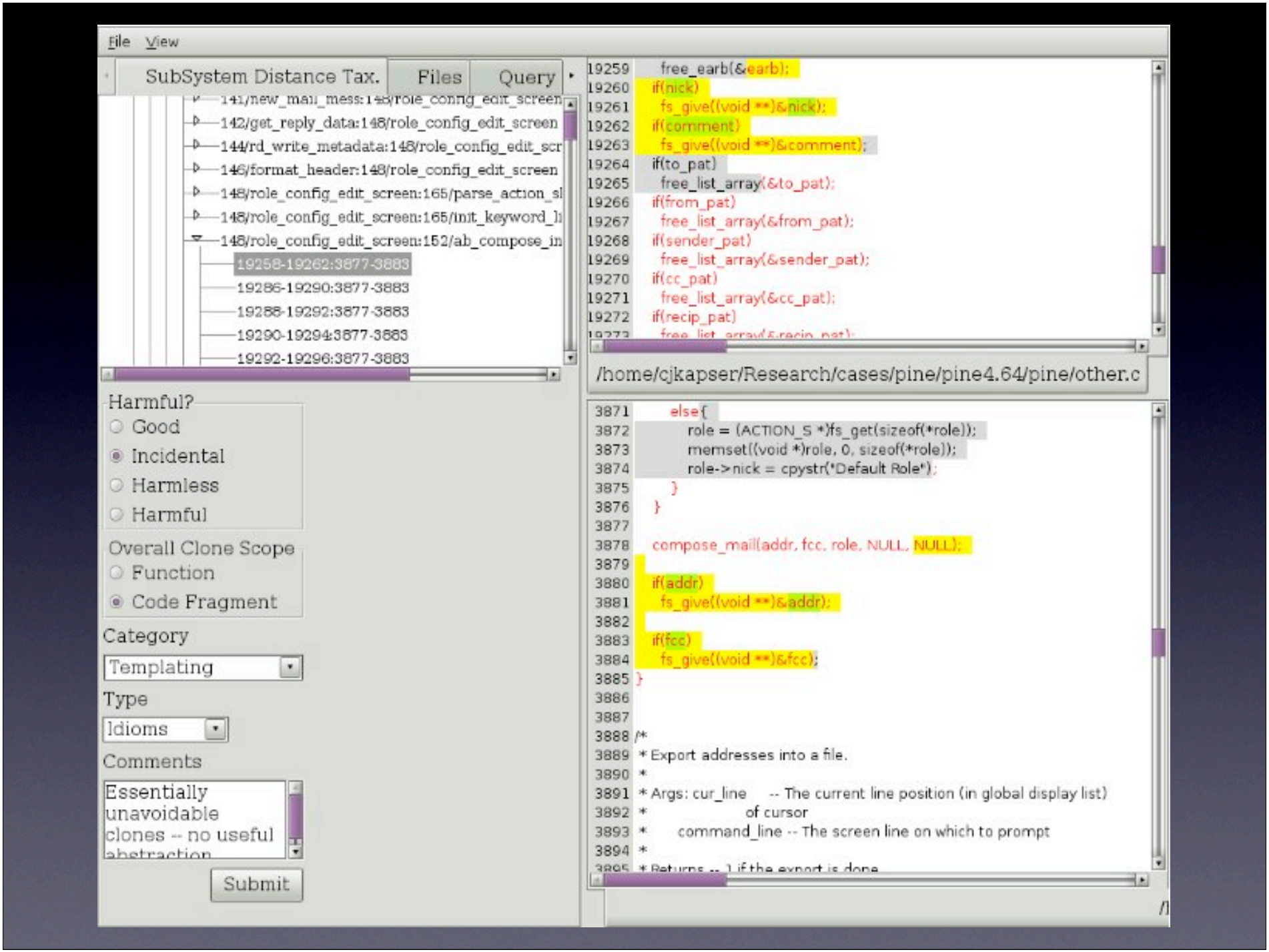
- Lots of work on open source systems ...
  - Linux, apache, gnumeric, PostgresQL
- ... but not so much on industrial practice
  - Does industry do it differently? How? Why?

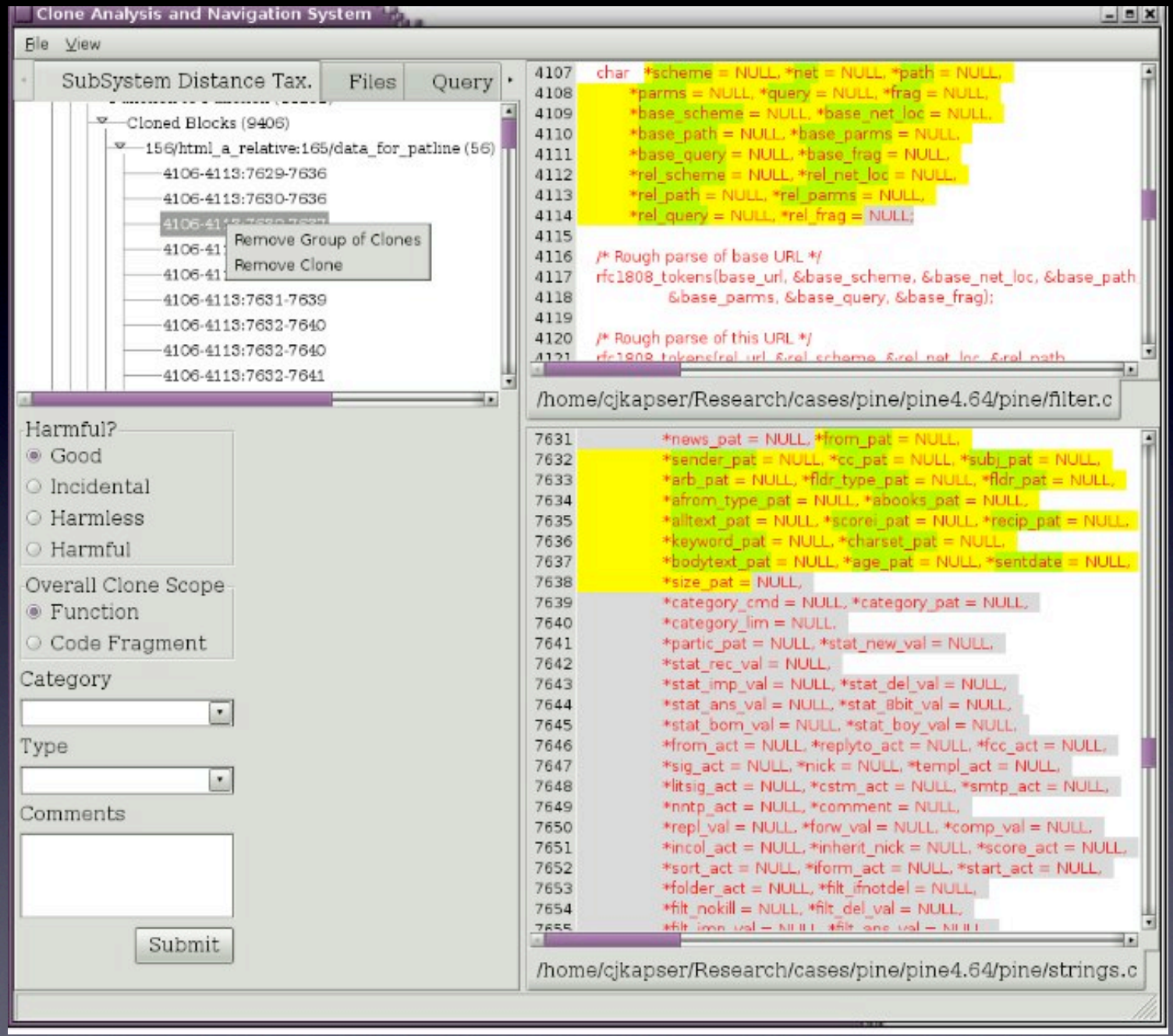
    e.g., forks vs. product lines

Goals of CLIP

- CLICS: A tool for clone detection + analysis
  - Better performance than CCfinder?
  - Remote detection, local analysis possible?
  - Contextual (not full source) browsing
  - Smart "taxonometric" support?


File View
SubSystem Distance Tax.
Files
Stats
Relations
Files
@PROGRAM
cjkapse
Research
cases
pine
pine4.64
build.cmd
contrib
doc
imap
docs
src
ansilib
c-client
charset
dmail
imapd
ipopd
mailutil
mlock
mtest
osdep
tmail
tools
pico
pine
Label
TYPE
Int. RCCs
Int. Clones
Ext. RCCs
Ext. Clones
Percentage of Clones
Percent of Lines
pine cSubSyst 7471 25757 923 1971 54.029518 57.626513
imap cSubSyst 10545 21931 1034 2119 46.852822 28.685179
pico cSubSyst 666 1198 27 49 2.429852 7.983043
contrib cSubSyst 81 171 233 359 1.032736 1.655205
doc cSubSyst 2 2 9 24 0.050863 3.967369
build.cmd cFile 0 0 0 0 0.000000 0.042691


File View
SubSystem Distance Tax.
Files
Query
141/new_mail_mess:148/role_config_edit_screen
142/get_reply_data:148/role_config_edit_screen
144/rd_write_metadata:148/role_config_edit_scr
146/format_header:148/role_config_edit_screen
148/role_config_edit_screen:165/parse_action_sl
148/role_config_edit_screen:165/init_keyword_li
148/role_config_edit_screen:152/ab_compose_in
19258-19262:3877-3883
19286-19290:3877-3883
19288-19292:3877-3883
19290-19294:3877-3883
19292-19296:3877-3883
Harmful?
Good
Incidental
Harmless
Harmful
Overall Clone Scope
Function
Code Fragment
Category
Templating
Type
Idioms
Comments
Essentially unavoidable clones -- no useful abstraction
Submit
19259 free_earb(&earb);
19260 if(nick)
19261 fs_give((void **)&nick);
19262 if(comment)
19263 fs_give((void **)&comment);
19264 if(to_pat)
19265 free_list_array(&to_pat);
19266 if(from_pat)
19267 free_list_array(&from_pat);
19268 if(sender_pat)
19269 free_list_array(&sender_pat);
19270 if(cc_pat)
19271 free_list_array(&cc_pat);
19272 if(recip_pat)
/home/cjkapser/Research/cases/pine/pine4.64/pine/other.c
3871 else{
3872 role = (ACTION_S *)fs_get(sizeof(*role));
3873 memset((void *)role, 0, sizeof(*role));
3874 role->nick = cpystr("Default Role");
3875 }
3876 }
3877
3878 compose_mail(addr, fcc, role, NULL, NULL);
3879
3880 if(addr)
3881 fs_give((void **)&addr);
3882
3883 if(fcc)
3884 fs_give((void **)&fcc);
3885 }
3886
3887
3888 /*
3889 * Export addresses into a file.
3890 *
3891 * Args: cur_line -- The current line position (in global display list)
3892 * of cursor
3893 * command_line -- The screen line on which to prompt
3894 *


Clone Analysis and Navigation System
File View
SubSystem Distance Tax.
Files
Query
Cloned Blocks (9406)
156/html_a_relative:165/data_for_patline (56)
4106-4113:7629-7636
4106-4113:7630-7636
Remove Group of Clones
Remove Clone
4106-4113:7631-7639
4106-4113:7632-7640
4106-4113:7632-7640
4106-4113:7632-7641
Harmful?
Good
Incidental
Harmless
Harmful
Overall Clone Scope
Function
Code Fragment
Category
Type
Comments
Submit
4107 char *scheme = NULL, *net = NULL, *path = NULL,
4108 *parms = NULL, *query = NULL, *frag = NULL,
4109 *base_scheme = NULL, *base_net_loc = NULL,
4110 *base_path = NULL, *base_parms = NULL,
4111 *base_query = NULL, *base_frag = NULL,
4112 *rel_scheme = NULL, *rel_net_loc = NULL,
4113 *rel_path = NULL, *rel_parms = NULL,
4114 *rel_query = NULL, *rel_frag = NULL;
4115
4116 /* Rough parse of base URL */
4117 rfc1808_tokens(base_url, &base_scheme, &base_net_loc, &base_path
4118 &base_parms, &base_query, &base_frag);
4119
4120 /* Rough parse of this URL */
/home/cjkapser/Research/cases/pine/pine4.64/pine/filter.c
7631 *news_pat = NULL, *from_pat = NULL,
7632 *sender_pat = NULL, *cc_pat = NULL, *subj_pat = NULL,
7633 *arb_pat = NULL, *fldr_type_pat = NULL, *fldr_pat = NULL,
7634 *afrom_type_pat = NULL, *abooks_pat = NULL,
7635 *alltext_pat = NULL, *score_pat = NULL, *recip_pat = NULL,
7636 *keyword_pat = NULL, *charset_pat = NULL,
7637 *bodytext_pat = NULL, *age_pat = NULL, *sentdate = NULL,
7638 *size_pat = NULL,
7639 *category_cmd = NULL, *category_pat = NULL,
7640 *category_lim = NULL,
7641 *partic_pat = NULL, *stat_new_val = NULL,
7642 *stat_rec_val = NULL,
7643 *stat_imp_val = NULL, *stat_del_val = NULL,
7644 *stat_ans_val = NULL, *stat_8bit_val = NULL,
7645 *stat_bom_val = NULL, *stat_boy_val = NULL,
7646 *from_act = NULL, *replyto_act = NULL, *fcc_act = NULL,
7647 *sig_act = NULL, *nick = NULL, *templ_act = NULL,
7648 *litsig_act = NULL, *cstm_act = NULL, *smtp_act = NULL,
7649 *nntp_act = NULL, *comment = NULL,
7650 *repl_val = NULL, *forw_val = NULL, *comp_val = NULL,
7651 *incol_act = NULL, *inherit_nick = NULL, *score_act = NULL,
7652 *sort_act = NULL, *iform_act = NULL, *start_act = NULL,
7653 *folder_act = NULL, *filt_ifnotdel = NULL,
7654 *filt_nokill = NULL, *filt_del_val = NULL,
/home/cjkapser/Research/cases/pine/pine4.64/pine/strings.c

Goals of CLIP

- Patterns of industrial cloning
  - Build on taxonomy of '*Cloning considered harmful' considered harmful* *[WCRE-06, EMSE-09]*
  - Rationale, short- vs. long-term effects, evolution, management, ...
  - Longitudinal statistical evaluation of cost/ benefit of (different kinds of) cloning

Goals of CLIP

- Improving industrial practice: assessment and management
  - — Quick analysis of new source (M&A)
  - — Auto-markup of risky cloning
  - — Linked editing

Open questions

- How important is feedback from original developers on design rationale of systems?
  - How easy will it be to obtain?
- How can we make the results useful to developers?
  - If tools are produced, will they be used?

CLIP
CLoning as
Industrial Practice

A research project involving UWaterloo and CA