Methodical Accelerator Design
Project Status Report

Laurent Deniau
CERN-BE/ABP

1st November 2011

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Achievements 1/3

- June – September (3 effective weeks)
  - ➡ First contact with the code (not easy!)
  - ➡ Mailing lists created (`mad-pub`, `mad-usr`, `mad-dev@cern.ch`)
  - ➡ Repository reorganized (mad@lxplus account)
  - ➡ Web sites (`http://cern.ch/mad`, `mad8`, `mad9`, `madx`)
  - ➡ Few bugs identified and corrected
  - ➡ Build process effective for Linux, MacOS X and Windows, 32 & 64 bits
  - ➡ 3 development releases 5.00.06 - 5.00.08

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Achievements 2/3

- October (3 effective weeks)
  - ➡ Project draft proposal for MAD-X Evolution in 3 phases (26 p.)
    - under discussion, should be significantly reorganized (not urgent)
      1) improve the code quality, make a build system, no change to the observable behavior
      2) improve architecture and design (should be removed, how-to is not a concern)
      3) improve the physics (should be rewritten, how-to is not a concern)
    - should only focus on observable objectives and metric of achievements
  - ➡ Provide access to the command line arguments
    - MAD-X `main` back in C
  - ➡ Extensions .F90 moved to .f90 for consistency
    - `-fpp` or `-Cpp` options added instead, only used for a very specific case
  - ➡ Hundred of functions signatures corrected
    - `foo()` to `foo(void)`
    - ≈ 20 functions had inconsistent signatures (caller ≠ callee)
    - probably solved few stack frame corruptions but not all
  - ➡ Update the `Makefile` according to changes (temporary)

Achievements 3/3

- October (3 effective weeks) *cont.*
  - ➡ Remove C and Fortran I/O wrappers and Python scripts (fragile and broken)
    - make `stdout` unbuffered instead (no performance loss measured for ~100k output)
  - ➡ Improve few string manipulation functions $O(n^2)$ ⇨ $O(n)$
    - improve speed by 10-15% on typical madx scripts runs
  - ➡ Split 10 piled files into 66 separate “modules” to increase the cohesion (not easy!)
    - not perfect (files `mad_*.[hc]`, temporary split and names)
    - need to be closed, ongoing, takes time, **only 40-60% is possible**
    - **no change to the semantic!** (identical digits by digits)

```
mad_6track.c    mad_core.h      mad_elemrfc.h   mad_ibs.h       mad_node.c      mad_regex.h     mad_time.c
mad_6track.h    mad_cst.c       mad_emit.c      mad_logic.c     mad_node.h      mad_rpn.c       mad_time.h
mad_aper.c      mad_cst.h       mad_emit.h      mad_logic.h     mad_option.c    mad_rpn.h       mad_touschek.c
mad_aper.h      mad_def.h       mad_err.c       mad_macro.c     mad_option.h    mad_select.c    mad_touschek.h
mad_api.c       mad_dict.c      mad_err.h       mad_macro.h     mad_orbit.c     mad_select.h    mad_track.c
mad_api.h       mad_dict.h      mad_eval.c      mad_main.c      mad_orbit.h     mad_seq.c       mad_track.h
mad_array.c     mad_dynap.c     mad_eval.h      mad_main.h      mad_out.c       mad_seq.h       mad_twiss.c
mad_array.h     mad_dynap.h     mad_exec.c      mad_match.c     mad_out.h       mad_sodd.c      mad_twiss.h
mad_beam.c      mad_elem.c      mad_exec.h      mad_match.h     mad_parse.c     mad_sodd.h      mad_util.c
mad_beam.h      mad_elem.h      mad_expr.c      mad_match2.c    mad_parse.h     mad_str.c       mad_util.h
mad_cmd.c       mad_elemdrift.c mad_expr.h      mad_match2.h    mad_plot.c      mad_str.h       mad_var.c
mad_cmd.h       mad_elemdrift.h mad_gcst.c      mad_math.c      mad_plot.h      mad_stream.c    mad_var.h
mad_cmdin.c     mad_elemerr.c   mad_gcst.h      mad_math.h      mad_ptc.c       mad_stream.h    mad_vec.c
mad_cmdin.h     mad_elemerr.h   mad_gfun.h      mad_mem.c       mad_ptc.h       mad_survey.c    mad_vec.h
mad_cmdpar.c    mad_elemmultp.c mad_gvar.c      mad_mem.h       mad_rand.c      mad_survey.h    mad_wrap_f.h
mad_cmdpar.h    mad_elemmultp.h mad_gvar.h      mad_mkthin.c    mad_rand.h      mad_sxf.c
mad_const.c     mad_elemprobe.c mad_help.c      mad_mkthin.h    mad_range.c     mad_sxf.h
mad_const.h     mad_elemprobe.h mad_help.h      mad_name.c      mad_range.h     mad_table.c
mad_core.c      mad_elemrfc.c   mad_ibs.c       mad_name.h      mad_regex.c     mad_table.h
```


Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Next Steps

- November – January
  - ➡ Classify and document the bugs (ongoing), not all are solvable...
    - the information should help to take decision on initiatives
  - ➡ Close as much as possible the “modules”. It will take several weeks (months?)
    - increase the cohesion (should be possible to a reasonable level)
    - decrease the coupling (will NOT be possible to a reasonable level)
  - ➡ Rewrite the main web page (update contain, change the structure)
  - ➡ Rewrite the proposal after identifying better observable objectives:

    1) improve the physics of the Twiss module (TBD)

    2) improve the physics of the Makethin module (TBD)

    3) improve the scripting language robustness (TBD)

    4) remove pending bugs not solvable with the current architecture

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch

Epilogue: example of “unsolvable” bug

```
struct node* new_node(char* name)
{
  char rout_name[] = "new_node";
  struct node* p = mycalloc(rout_name,1, sizeof(struct node));
  strcpy(p->name, name);
  p->stamp = 123456;
  if (watch_flag) fprintf(debug_file, "creating ++> %s\n", p->name);
  return p;
}

struct node* clone_node(struct node* p, int flag)
{
  struct node* clone = new_node(p->name);
  strcpy(clone->name,p->name);
  clone->base_name = p->base_name;
  clone->occ_cnt = p->occ_cnt;
  clone->sel_err = p->sel_err;
  clone->position = p->position;
  clone->at_value = p->at_value;
  clone->length = p->length;
  clone->at_expr = p->at_expr;
  clone->from_name = p->from_name;
  clone->p_elem = p->p_elem;
  clone->p_sequ = p->p_sequ;
  clone->savebeta = p->savebeta;
  if (flag)
  {
    clone->p_al_err = p->p_al_err;
    clone->p_fd_err = p->p_fd_err;
  }
  return clone;
}

struct node* delete_node(struct node* p)
{
  char rout_name[] = "delete_node";
  if (p == NULL) return NULL;
  if (stamp_flag && p->stamp != 123456)
    fprintf(stamp_file, "d_n double delete --> %s\n", p->name);
  if (watch_flag) fprintf(debug_file, "deleting --> %s\n", p->name);
  if (p->p_al_err) p->p_al_err = delete_double_array(p->p_al_err);
  if (p->p_fd_err) p->p_fd_err = delete_double_array(p->p_fd_err);
  myfree(rout_name, p);
  return NULL;
}
```

Incomplete initialization
Assumes all bits = 0 means value = 0
Should not be used alone
User responsability!

Duplicated code

Not a clone (hybrid shallow copy)!
User responsability!

Not symmetric (sxf)
User responsability!

Laurent Deniau, CERN BE/ABP, 1211 Geneva 23, laurent.deniau@cern.ch