Sirepo: Low Code Control System GUIs

Evan Carlin*, Paul Moeller, Gurhar Khalsa, Mike Keilman, Jon Edelen, Robert Nagler

*evan@radiasoft.net

October 7, 2023 23, 2022

ICALEPCS Controls GUI Workshop, Cape Town ZA

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Science, Office of Basic Energy Sciences under Award Number DE-SC0021551.

Introduction

- Evan Carlin
  - Software Group Leader at RadiaSoft
- RadiaSoft (https://www.radiasoft.net/)
  - Software company in Boulder, CO that specializes in R&D and consulting for beamline physics
- Sirepo
  - Browser based gateway that provides GUIs around legacy codes for particle physics simulations, 3D visualization, tool for designing ML models, a Jupyter server, and recently a way to connect to controls system (https://www.sirepo.com)
  - It wears many hats => it is a framework
  - Open source https://github.com/radiasoft/sirepo

Overarching GUI Themes

- Start with the browser
  - People are familiar with using it
  - They are fast enough for the tasks we encounter
  - Familiar development technologies that integrate well with other systems (ex authentication)
  - Applications are easy to update
- Allow users of different skill levels
  - Power users need to be supported along with beginners
  - Find your user and tailor the GUI to them. Expose "advanced settings" behind buttons
  - Give the most advanced users the ability to download the underlying code, data, etc
- Make the deployments easy too
  - GUIs that involve low code / no code generally need to be deployed in an easy manner as well

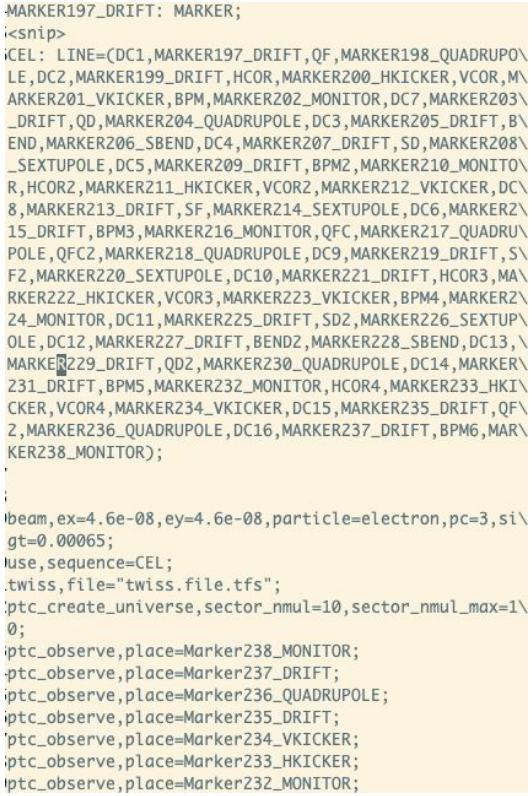
Demo

- Start with a MAD-X input file
  - MAD (Methodical Accelerator Design) is a popular tool for the modeling of particle accelerators
  - Provides a DSL to specify a lattice, initial conditions, and run simulations
- Upload to sirepo.com/madx
  - Lattice is imported and visualized
  - You can edit it and run a simulation
  - You can build a lattice without uploading a MAD-X input file
- Import into sirepo.com/controls
  - See process variables
  - Run optimization
  - Connect to live-system to monitor

Demo

Common data model

- To go from MAD-X input file, to MAD-X Sirepo app, to Controls app (and back) we have a common data model that underpins our system



```
MARKER197_DRIFT: MARKER;
<snip>
CEL: LINE=(DC1,MARKER197_DRIFT,QF,MARKER198_QUADRUPO\
LE,DC2,MARKER199_DRIFT,HCOR,MARKER200_HKICKER,VCOR,M\
ARKER201_VKICKER,BPM,MARKER202_MONITOR,DC7,MARKER203\
_DRIFT,QD,MARKER204_QUADRUPOLE,DC3,MARKER205_DRIFT,B\
END,MARKER206_SBEND,DC4,MARKER207_DRIFT,SD,MARKER208\
_SEXTUPOLE,DC5,MARKER209_DRIFT,BPM2,MARKER210_MONITO\
R,HCOR2,MARKER211_HKICKER,VCOR2,MARKER212_VKICKER,DC\
8,MARKER213_DRIFT,SF,MARKER214_SEXTUPOLE,DC6,MARKER2\
15_DRIFT,BPM3,MARKER216_MONITOR,QFC,MARKER217_QUADRU\
POLE,QFC2,MARKER218_QUADRUPOLE,DC9,MARKER219_DRIFT,S\
F2,MARKER220_SEXTUPOLE,DC10,MARKER221_DRIFT,HCOR3,MA\
RKER222_HKICKER,VCOR3,MARKER223_VKICKER,BPM4,MARKER2\
24_MONITOR,DC11,MARKER225_DRIFT,SD2,MARKER226_SEXTUP\
OLE,DC12,MARKER227_DRIFT,BEND2,MARKER228_SBEND,DC13,\
MARKER229_DRIFT,QD2,MARKER230_QUADRUPOLE,DC14,MARKER\
231_DRIFT,BPM5,MARKER232_MONITOR,HCOR4,MARKER233_HKI\
CKER,VCOR4,MARKER234_VKICKER,DC15,MARKER235_DRIFT,QF\
2,MARKER236_QUADRUPOLE,DC16,MARKER237_DRIFT,BPM6,MAR\
KER238_MONITOR);

beam,ex=4.6e-08,ey=4.6e-08,particle=electron,pc=3,si\
gt=0.00065;
use,sequence=CEL;
twiss,file="twiss.file.tfs";
ptc_create_universe,sector_nmul=10,sector_nmul_max=1\
0;
ptc_observe,place=Marker238_MONITOR;
ptc_observe,place=Marker237_DRIFT;
ptc_observe,place=Marker236_QUADRUPOLE;
ptc_observe,place=Marker235_DRIFT;
ptc_observe,place=Marker234_VKICKER;
ptc_observe,place=Marker233_HKICKER;
ptc_observe,place=Marker232_MONITOR;
```

MAD-X input file

```
{
  "models": {
    "beamlines": [
      {
        "angle": 0.369599135716446,
        "count": 68,
        "distance": 11.583932593720583,
        "id": 185,
        "items": [
          101,
          143,
          <snip>
        ],
        "length": 11.691414678740419,
        "name": "CEL"
      }
    ],
    "elements": [
      {
        "_id": 101,
        "l": 1.53667,
        "name": "DC1",
        "type": "DRIFT"
      },
      <snip>
    ],
    "commands": [
      {
        "_id": 186,
        "_super": "_COMMAND",
        "_type": "beam",
        "bcurrent": 0,
        "beta": 0,
        "brho": 0,
        "bunched": "1",
        "bv": 1,
```

Sirepo data model for MAD-X

```
{
  "models": {
    <snip>
    "controlSettings": {
      "bpmPlotSize": 0.0021,
      "defaultFactor": 100,
      "deviceServerMachine": "",
      "deviceServerProcId": "",
      "deviceServerProcName": "",
      "deviceServerURL": "",
      "deviceServerUser": "",
      "inputLogFile": "",
      "operationMode": "madx",
      "particlePlotSize": 0.1,
      "processVariables": [
        {
          "elId": 143,
          "isWritable": "0",
          "pvDimension": "horizontal",
          "pvName": ""
        },
      "readOnly": "1",
      "selectedTimeIndex": 0,
      "simMode": "singleUpdates"
    },
    "externalLattice": {
      "models": {
        "beamlines": [
          {
            "angle": 0.369599135716446,
            "count": 68,
            "distance": 11.583932593720583,
            "id": 185,
            "items": [
              101,
```

Sirepo data model for Controls

Schema to data model


```
"command_beam": {
    "_super": ["_", "model", "_COMMAND"],
    "particle": ["PARTICLE", "ParticleType", "positron", "The name of particles in the beam."],
    "mass": ["MASS [GeV]", "RPNValue", 0.0005109989461, "The restmass of the particles in the beam."],
    "charge": ["CHARGE [qp]", "RPNValue", 1, "The electrical charge of the particles in the beam in units of the proton charge"],
    "energy": ["ENERGY [GeV]", "RPNValue", 0.0, "Total energy per particle."],
    "pc": ["PC [GeV/c]", "RPNValue", 0.0, "Momentum per particle."],
    "gamma": ["GAMMA", "RPNValue", 0.0, "Relativistic factor, ie the ratio between total energy and rest energy of the particles."],
    "beta": ["BETA [v/c]", "RPNValue", 0.0, "Ratio between the speed of the particles and the speed of light."],
    "brho": ["BRHO [P/abs(q)]", "RPNValue", 0.0, "Magnetic rigidity of the particles."],
    "ex": ["EX [m]", "RPNValue", 1.0, "The horizontal emittance."],
    "exn": ["EXN", "RPNValue", 0.0, "The normalised horizontal emittance."],
    "ey": ["EY [m]", "RPNValue", 1.0, "The vertical emittance."],
    "eyn": ["EYN", "RPNValue", 0.0, "The normalised vertical emittance."],
    "et": ["ET [m]", "RPNValue", 0.001, "The longitudinal emittance $\\epsilon_t$"],
    "sequence": ["SEQUENCE", "OptionalLatticeBeamlineList", "", "Attaches the beam command to a specific sequence."],
    "sigt": ["SIGT [m]", "RPNValue", 1.0, "The bunch length $c\\space \\sigma_t$."],
    "sige": ["SIGE", "RPNValue", 0.001, "The relative energy spread $\\sigma_E / E$."],
    "kbunch": ["KBUNCH", "Integer", 1, "The number of particle bunches in the machine."],
    "npart": ["NPART", "Integer", 1, "The number of particles per bunch."],
    "bcurrent": ["BCURRENT", "RPNValue", 0.0, "The bunch current."],
    "bunched": ["BUNCHED", "Boolean", "1", "The beam is treated as bunched whenever this makes sense."],
```


MAD-X Schema


```
{
    "models": {
        "beamlines": [
            {
                "angle": 0.369599135716446,
                "count": 68,
                "distance": 11.583932593720583,
                "id": 185,
                "items": [
                    101,
                    143,
                    <snip>
                ],
                "length": 11.691414678740419,
                "name": "CEL"
            }
        ],
        "elements": [
            {
                "_id": 101,
                "l": 1.53667,
                "name": "DC1",
                "type": "DRIFT"
            },
            <snip>
        ],
        "commands": [
            {
                "_id": 186,
                "_super": "_COMMAND",
                "_type": "beam",
                "bcurrent": 0,
                "beta": 0,
                "brho": 0,
                "bunched": "1",
                "bv": 1,
```


MAD-X data model

Schema in detail

- Models
  - Define a "type" which encapsulates some data

```
"KICKER": {
    "_super": ["_", "model", "_APERTURE_ELEMENT"],
    "l": ["L [m]", "RPNValue", 0.0, "The length of the closed orbit corrector."],
    "hkick": ["HKICK", "RPNValue", 0.0, "The horizontal momentum change $\\delta PX$"],
    "vkick": ["VKICK", "RPNValue", 0.0, "The vertical momentum change $\\delta PY$"],
    "tilt": ["TILT [rad]", "RPNValue", 0.0, "The roll angle about the longitudinal axis."]
},
```

- Views
  - Define a GUI layout

```
"bunchReport": {
    "title": "Bunch Report",
    "advanced": [
        [
            ["Horizontal",
                ["x"]
            ],
            ["Vertical",
                ["y"]
            ]
        ],
        "histogramBins",
        "colorMap",
        "notes"
    ],
    "basic": []
},
```

GUI generation and app specific code

- Generalized components

```
SIREPO.app.directive('fieldEditor', function(appState, keypressService, panelState, utilities) {
    return {
        restrict: 'A',
        scope: {
            modelName: '=',
            field: '=fieldEditor',
            model: '=',
            viewName: '=',
            <snip>
        },
        template: `
            <div data-ng-class="utilities.modelFieldID(modelName, field)">
            <div data-ng-show="showLabel" data-label-with-tooltip="" class="control-label" data-ng-class="labelClass" data-label="{{ customLabel || info[0] }}" data-tooltip="{{ info[3] }}"></div>
            <div data-ng-switch="info[1]">
              <div data-ng-switch-when="Integer" data-ng-class="fieldClass">
                <input data-string-to-number="integer" data-ng-model="model[field]" data-min="info[4]" data-max="info[5]" class="form-control" style="text-align: right" data-lpignore="true" required />
              </div>
              <div data-ng-switch-when="Float" data-ng-class="fieldClass">
                <input data-string-to-number="" data-ng-model="model[field]" data-min="info[4]" data-max="info[5]" class="form-control" style="text-align: right" data-lpignore="true" required />
                <div class="sr-input-warning"></div>
              </div>
```

- Mixing in app specific behavior

```
SIREPO.app.config(() => {
    SIREPO.SINGLE_FRAME_ANIMATION = ['beamPositionAnimation', 'instrumentAnimationTwiss'];
    SIREPO.appFieldEditors += `
        <div data-ng-switch-when="MadxSimList" data-ng-class="fieldClass">
          <div data-sim-list="" data-model="model" data-field="field" data-code="madx" data-route="lattice"></div>
        </div>
        <div data-ng-switch-when="AmpTable">
          <div data-amp-table=""></div>
        </div>
        <div data-ng-switch-when="AmpField">
          <div data-amp-field=""></div>
        </div>
        <div data-ng-switch-when="ProcessVariables" class="col-sm-12">
          <div element-pv-fields=""></div>
        </div>
    `;
```

Deployment

- We work with many customers
  - We need to be able to deploy quickly/easily to systems we do not have much control over (no root access)
- Need to manage dependency chaos
  - Each deployment target (lab, beamline, individual) has their own unique dependencies
- Monolithic application bundled in Docker
  - Releases can be as simple as a `docker pull`
  - We "control our world". We need to make sure our code works inside of a docker container we control

Thanks! Questions?

Job system

- Manages processes using resources for users
- Initially this was designed for managing simulations (ex a user gets one agent with 4 cores and one agent with 1 core)
- The system was extended so agents that previously ran simulations are now connected to controls systems