Building EPICS Support Modules with SUMO

Benjamin Franksen, Götz Pfeiffer

Helmholtz-Zentrum Berlin für Materialien und Energie

EPICS Meeting Spring 2019

The Problem

- suppose...
    - your application depends on a lot of support modules
    - something deep down in the dependency graph changes (base, asyn, sequencer,...)
    - caused by a patch/hotfix, or an upgrade to a new version
- need to re-build and re-install all dependent modules
- doing this manually is tedious and error-prone:
    - change `configure/RELEASE` for each module
    - ensure correct order (consistency)
    - don't forget to rebuild anything (completeness)
    - invent a name (path) for the new builds so we don't break existing applications (persistence)

The Solution

- what it should do
  1. automatically download, build, and install modules
  2. distinguish builds of the same module version against different versions of its dependencies
  3. operate non-destructively: existing builds remain available for use by other applications
  4. end-application should define versions of every support module it depends on
- what we need for that to work
  1. replace application's `configure/RELEASE` with something more abstract that declares module versions (`configure/MODULES`)
  2. an external database of module versions with information about dependencies and source locations

Example of a configure/MODULES file

```
{
    "alias": [
        "BASE:EPICS_BASE",
        "SEQ:SNCSEQ"
    ],
    "module": [
        "ASYN:R4-32-bessy1",
        "AUTOSAVE:R4-8-bessy4",
        "BASE:R3-14-12-7-bessy2",
        "DEVIOCSTATS:R3-1-9-bessy5",
        "MOTOR:R6-9-bessy5",
        "SEQ:R2-2-5"
    ]
}
```

Module version database

example of an entry in the dependency database

```
> sumo db show SEQ:R2-2-5
{
    "SEQ": {
        "R2-2-5": {
            "aliases": {
                "BASE": "EPICS_BASE"
            },
            "dependencies": [
                "BASE"
            ],
            "source": {
                "darcs": {
                    "tag": "R2-2-5",
                    "url": "http://repo.acc.bessy.de/darcs/epics/
                        support/seq/branch-2-2"
                }
            }
        }
    }
}
```

Module version database

index: module name

```
> sumo db show SEQ:R2-2-5
{
    "SEQ": {
        "R2-2-5": {
            "aliases": {
                "BASE": "EPICS_BASE"
            },
            "dependencies": [
                "BASE"
            ],
            "source": {
                "darcs": {
                    "tag": "R2-2-5",
                    "url": "http://repo.acc.bessy.de/darcs/epics/
                        support/seq/branch-2-2"
                }
            }
        }
    }
}
```

Module version database

subindex: module version

```
> sumo db show SEQ:R2-2-5
{
    "SEQ": {
        "R2-2-5": {
            "aliases": {
                "BASE": "EPICS_BASE"
            },
            "dependencies": [
                "BASE"
            ],
            "source": {
                "darcs": {
                    "tag": "R2-2-5",
                    "url": "http://repo.acc.bessy.de/darcs/epics/
                        support/seq/branch-2-2"
                }
            }
        }
    }
}
```

Module version database

modules we depend on

```
> sumo db show SEQ:R2-2-5
{
    "SEQ": {
        "R2-2-5": {
            "aliases": {
                "BASE": "EPICS_BASE"
            },
            "dependencies": [
                "BASE"
            ],
            "source": {
                "darcs": {
                    "tag": "R2-2-5",
                    "url": "http://repo.acc.bessy.de/darcs/epics/
                        support/seq/branch-2-2"
                }
            }
        }
    }
}
```

Module version database

optional aliases (name to use in configure/RELEASE)

```
> sumo db show SEQ:R2-2-5
{
    "SEQ": {
        "R2-2-5": {
            "aliases": {
                "BASE": "EPICS_BASE"
            },
            "dependencies": [
                "BASE"
            ],
            "source": {
                "darcs": {
                    "tag": "R2-2-5",
                    "url": "http://repo.acc.bessy.de/darcs/epics/
                        support/seq/branch-2-2"
                }
            }
        }
    }
}
```

Module version database

where and how to get the sources

```
> sumo db show SEQ:R2-2-5
{
    "SEQ": {
        "R2-2-5": {
            "aliases": {
                "BASE": "EPICS_BASE"
            },
            "dependencies": [
                "BASE"
            ],
            "source": {
                "darcs": {
                    "tag": "R2-2-5",
                    "url": "http://repo.acc.bessy.de/darcs/epics/
                        support/seq/branch-2-2"
                }
            }
        }
    }
}
```

Source definitions

- either URL of tar ball or directory
- URL for a version control repository
  - plus tag/revision/commit (whatever uniquely identifies a version)
  - supported: CVS, Darcs, Git, Mercurial, Subversion
- can also add patch files:

```
"patches": [
    "/path/to/patchfile",
    ...
]
```

- we use mostly Darcs repos on a local server, even for modules we do not maintain
  - easy to maintain local patches
  - or configuration (e.g. in base)
- choose whatever model suits you best

Building support modules for an application

```
> sumo build new --makeflags="-sj"
creating build with tag 'BL-006'
[...residual build output...]
```

- create a unique fresh *build identifier* (here `BL-006`)
- for each `MODULE:VERSION` in `configure/MODULES`:
    - if the *same version* has already been built using the *same versions* for all its dependencies, then re-use that build for these modules
    - otherwise build the dependencies (recursively) and then the module itself with the new build name appended to the path
- store information about the build in the *build database*
- works by creating a Makefile
- parallel builds are supported (via `--makeflags` option)

Generating `configure/RELEASE` for the application

```
> sumo build use
using build BL-006
```

- search for a build that contains every module in exactly the version declared in `configure/MODULES`
- if one is found, generate `configure/RELEASE`
- otherwise fail with an error message

Application's configure/RELEASE

```
> cat configure/RELEASE
# generated by sumo using build BL-006:
SNCSEQ=/opt/sumo/build/SEQ/R2-2-5+BL-006
MOTOR=/opt/sumo/build/MOTOR/R6-9-bessy5+BII-283
DEVIOCSTATS=/opt/sumo/build/DEVIOCSTATS/
  R3-1-9-bessy5+BII-283
AUTOSAVE=/opt/sumo/build/AUTOSAVE/R4-8-bessy4+BL-004
ASYN=/opt/sumo/build/ASYN/R4-32-bessy1+BL-004
EPICS_BASE=/opt/sumo/build/BASE/
  R3-14-12-7-bessy2+BII-283
```

identifier for this build

Application's configure/RELEASE

```
> cat configure/RELEASE
# generated by sumo using build BL-006:
SNCSEQ=/opt/sumo/build/SEQ/R2-2-5+BL-006
MOTOR=/opt/sumo/build/MOTOR/R6-9-bessy5+BII-283
DEVIOCSTATS=/opt/sumo/build/DEVIOCSTATS/
  R3-1-9-bessy5+BII-283
AUTOSAVE=/opt/sumo/build/AUTOSAVE/R4-8-bessy4+BL-004
ASYN=/opt/sumo/build/ASYN/R4-32-bessy1+BL-004
EPICS_BASE=/opt/sumo/build/BASE/
  R3-14-12-7-bessy2+BII-283
```

variable name and module name differ due to `alias` definitions

Application's configure/RELEASE

```
> cat configure/RELEASE
# generated by sumo using build BL-006:
SNCSEQ=/opt/sumo/build/SEQ/R2-2-5+BL-006
MOTOR=/opt/sumo/build/MOTOR/R6-9-bessy5+BII-283
DEVIOCSTATS=/opt/sumo/build/DEVIOCSTATS/
  R3-1-9-bessy5+BII-283
AUTOSAVE=/opt/sumo/build/AUTOSAVE/R4-8-bessy4+BL-004
ASYN=/opt/sumo/build/ASYN/R4-32-bessy1+BL-004
EPICS_BASE=/opt/sumo/build/BASE/
  R3-14-12-7-bessy2+BII-283
```

existing build BII-283 is re-used here

Application's configure/RELEASE

```
> cat configure/RELEASE
# generated by sumo using build BL-006:
SNCSEQ=/opt/sumo/build/SEQ/R2-2-5+BL-006
MOTOR=/opt/sumo/build/MOTOR/R6-9-bessy5+BII-283
DEVIOCSTATS=/opt/sumo/build/DEVIOCSTATS/
  R3-1-9-bessy5+BII-283
AUTOSAVE=/opt/sumo/build/AUTOSAVE/R4-8-bessy4+BL-004
ASYN=/opt/sumo/build/ASYN/R4-32-bessy1+BL-004
EPICS_BASE=/opt/sumo/build/BASE/
  R3-14-12-7-bessy2+BII-283
```

same with build BL-004

Rationale for JSON

- easy to read and edit for humans
- easy to write third-party tools that query or update the module database e.g. sumo-edit (written in Haskell, https://www-csr.bessy.de/cgi-bin/darcsweb.cgi?r=sumo-edit)
- sorted and white-space formatted ⇒ meaningful diffs
- sumo keeps it under version control (if configured that way)
- also used for build database and configuration

Particular design decisions / Non-goals

- versions are *not* ordered
  - allowing version ranges requires the tool to make (arbitrary) choices
  - instead we require that the application defines exact versions
  - gives predictable and (ideally) reproducible results
- dependencies of a module version are *not* versioned
  - hard to claim compatibility with a range of versions a priori
  - if done conservatively: may unnecessarily restrict combinations
  - if done liberally: may give a false sense of safety
  - better to allow all combinations and let things fail
  - again, responsibility lies with the end-application
- data model is repetitive but simple and robust
- tool support for common operations (e.g. derive a new version from an existing one, changing only version name and tag)

Additional details

- configurable via command line options and/or JSON file
- command line syntax with nested subcommands
  - online help via `sumo help [command]`
  - powerful `bash` command line completion (works for `zsh`, too)
- integration with EPICS build rules

```
$(TOP)/configure/RELEASE: $(TOP)/configure/MODULES $(TOP)/
     sumo.config
    @echo "Re-creating $@ from $<"
    cd $(TOP) && sumo build use
```

  - automatically re-create `configure/RELEASE` whenever `configure/MODULES` changes
  - building fails if there is no suitable sumo build available

Status

- we have been using it continually since 2014
- works reliably, actively maintained, stable
- author and maintainer is my colleague Götz Pfeiffer
- extensively documented: https://goetzpf.bitbucket.io/sumo/
- written in Python (2 and 3)
- mercurial repo: https://bitbucket.org/goetzpf/epics-sumo
- tar files, RPMs, and Debian packages: https://bitbucket.org/goetzpf/epics-sumo/downloads/
- ... or just say

```
> pip install EPICS-sumo
```

Conclusion

- upgrading modules, applying patches, fixing bugs, or playing with modifications becomes painless
- even if the change is deep down in the dependency graph
- encourages factoring common parts of applications to small support modules, resulting in reduced code duplication
- the work of creating the initial dependency database quickly pays off

- I would *never* want go back to manually building EPICS support modules

Building things other EPICS support modules

- instead of calling `make`, do something different
- configured using `"make-recipes"` key
- whose value is a list of command line strings to put as the recipe into the generated `Makefile`
- we use it to build RTEMS, using a wrapper script

Additional tools: sumo-scan

- create an initial dependency database by scanning an existing tree of compiled support modules
- based on heuristics, thus
    - may not work for you if your directory layout differs significantly from what we had (before sumo)
    - needs tweaking and fiddling with the parameters
- can still be very useful to get started