PAUL SCHERRER INSTITUT

PSI

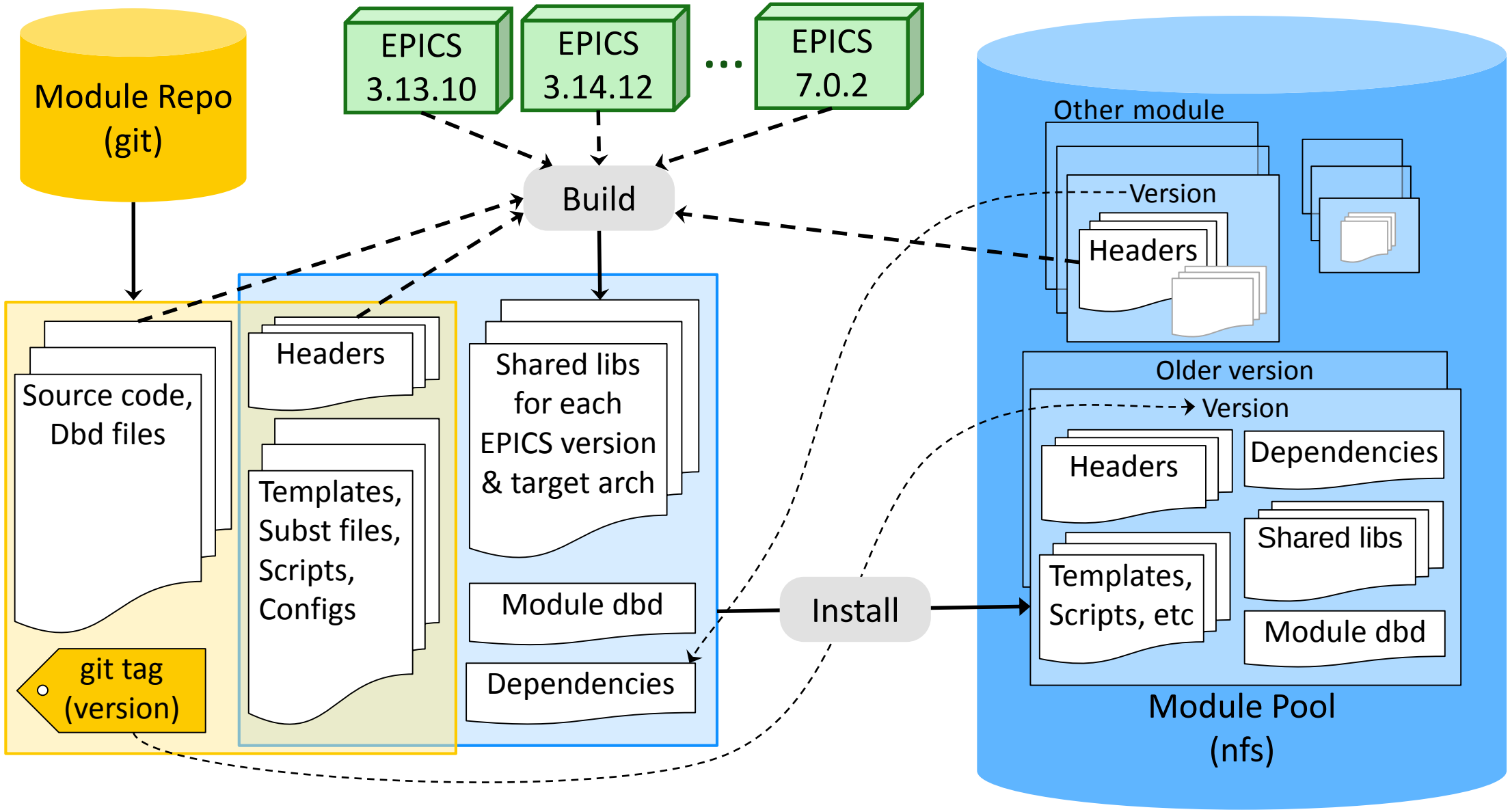
Dirk Zimoch :: Control Systems :: Paul Scherrer Institut

Modular builds at PSI

EPICS Collaboration Meeting ITER June 2019

PAUL SCHERRER INSTITUT

Building a Module



driver.makefile

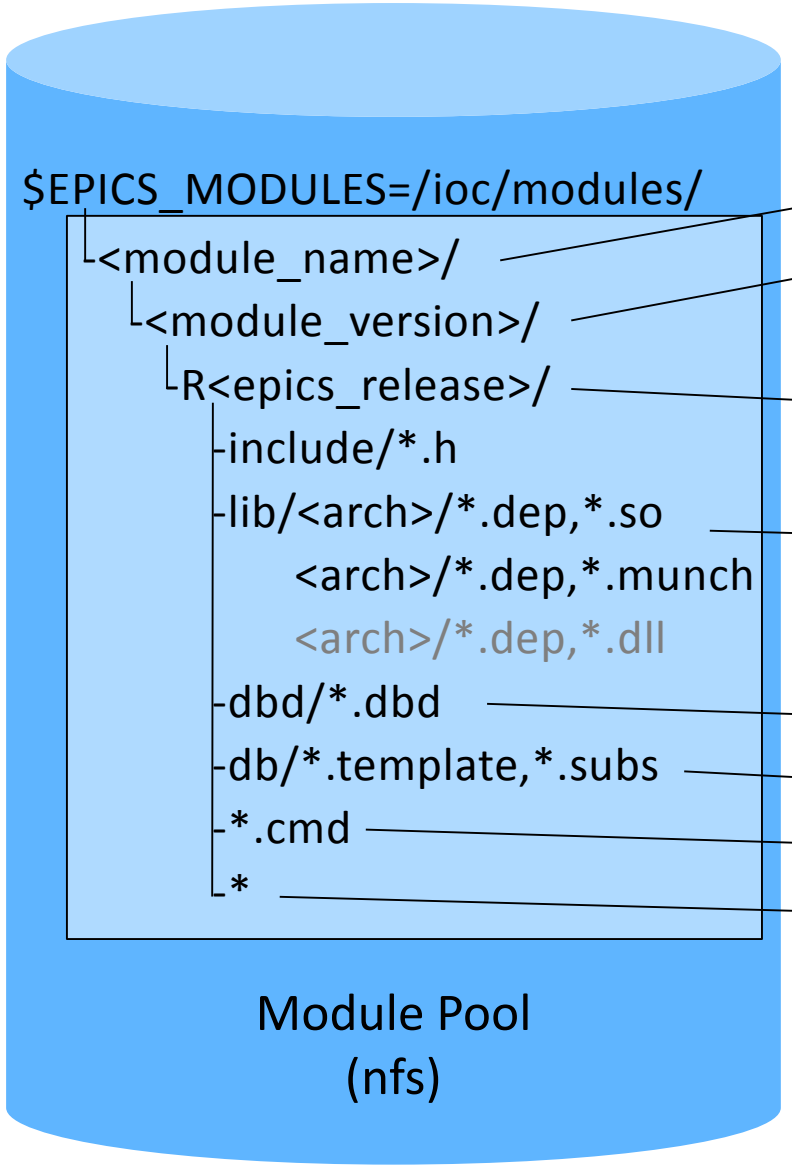
Makefile

```
include /ioc/toos/driver.makefile
EXCLUDE_VERSIONS = 3.13
BUILDCLASSES = Linux vxWorks
ARCH_FILTER = …
…
```

make [build, install, clean, uninstall]

- Automatically finds source, dbd, template, script files, but you can change this.
- Automatically finds headers of latest versions of already installed modules, but you can change this.
  - Creates dependency file with names and versions of modules whose headers have been used.
- Builds module for all EPICS versions and target architectures, but you can change this.
  - Uses build systems of installed EPICS versions.
- Creates module version from GIT (or CVS) tag.
  - Or uses username for (untagged) test versions, but you can change this.

Module Pool Structure

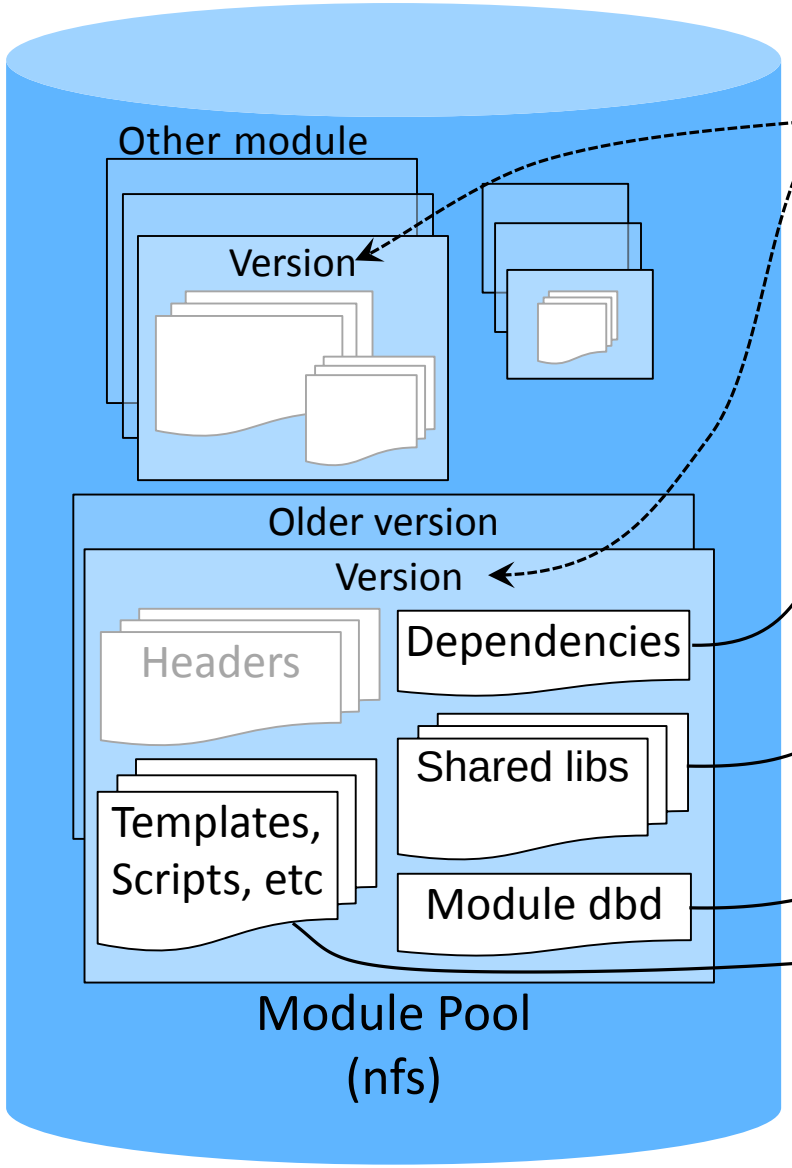


- Each module in its own directory
- Subdirectory per module version
  - numbers: `2.4.7` (released versions)
  - names: `zimoch` (test versions)
- Subdirectory per EPICS base version
  - R3.13.10, R3.14.12, …R7.0.2
- lib/directories for each target arch
  - dependency file
  - dynamically loadable library
- module dbd file
- templates and substitution files
- scripts (startup script snippets)
- arbitrary config files

Simple rollback: delete the directory

Loading a Module

```
require modulename [version] ["macro=value,…"]
```



1. Find matching version
2. Load dependencies recursively
3. Load library (lib<module>.so,
   <module>Lib.munch, <module>.dll)
4. Load dbd file <module>.dbd
5. Call <module>_registerRecordDeviceDriver
6. Set environment variables
   - MODULE
   - ${MODULE}_VERSION
   - ${MODULE}_DIR
   - ${MODULE}_DB
   - prepend to EPICS_DB_INCLUDE_PATH
   - prepend to SCRIPT_PATH
7. Execute startup script snippet (with macros)
   - may configure drivers
   - may load records
8. Load template with module version infos
9. Log loaded versions to a database

PAUL SCHERRER INSTITUT

Static Build vs Modular Build

Static

- IOC does not change without explicit action
  - Supporter has full control
  - Software may get outdated (bugs, features)
- Code is built for each IOC
  - No need to coordinate with others
  - No need to support archs/versions not used
  - Rollback is local to IOC
- Over-all heterogeneous system
  - More difficult for on-call supporters

Modular

- IOC updates automatically with every reboot
  - Bug fixes are automatically deployed
  - New features are available immediately
  - All IOCs run the current versions
- Code is built centrally
  - IOC supporter does not need to care about
    - make
    - EPICS base
    - compiler errors/warnings
  - Can improve code quality
- Need way to roll back if necessary
  - Log which versions had been used before
  - Allow to load older versions
  - Make uninstall simple