BUILD & DEPLOY TOOLS FOR ACCELERATOR CONTROL SYSTEMS AT HZB

HZB BESSY II Light Source

An Overview
... a short one
... honestly

Thomas Birke on behalf of the whole workgroup - BE-IA-AOT/ACS

2022-09-20 - Workshop on Build and Deployment
EPICS Meeting 2022 @ COSYLAB, Ljubljana, Slovenia

RESPONSIBILITIES OF BE-IA-AOT / ACS

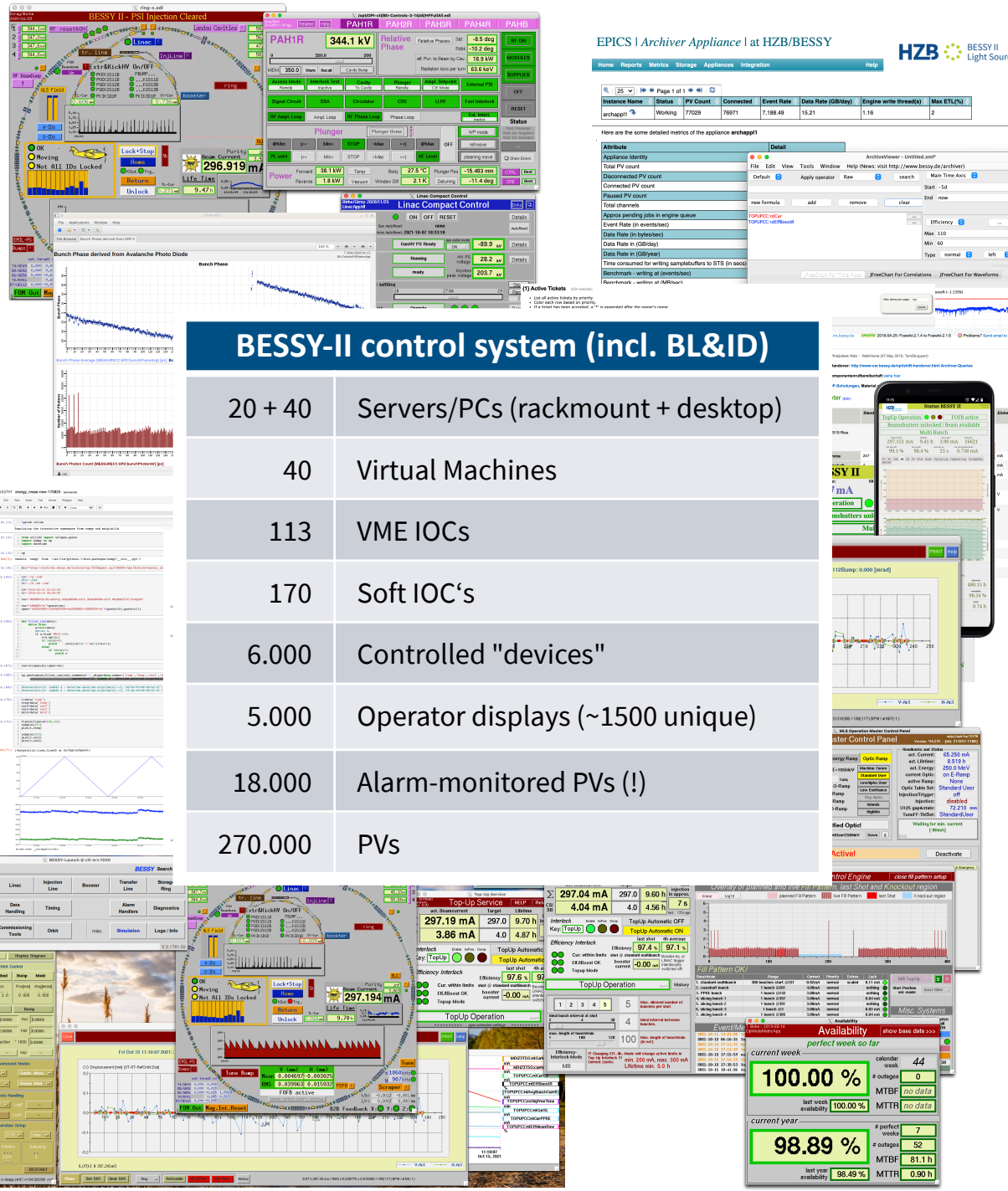
Provide everything necessary to control and operate HZB accelerators and all of their relevant components.

... except for PLC-based interlocks, diagnostics, timing...
and everything down the beamlines!

Group of 14 provides:

- **Device Integration**
- **Operator Interfaces**, Generic Tools, handling of Alarms…
- **Application Software:** Automation of Complex Procedures
- **Services**: Logging, Archiving, CA-Gateways, Databases, shift-schedule, elog, trac, wiki, live data web-displays, …
- **Operations** – shifts, on-call (component & supervisor)
- **Support Scientific Software Development**
  Accelerator Commissioning, -Optimization and -Research
- **Remote Access** Control-Room session (read-only!)
- Maintain our own **"Domain Specific IT"**

BESSY-II control system (incl. BL&ID)	
20 + 40	Servers/PCs (rackmount + desktop)
40	Virtual Machines
113	VME IOCs
170	Soft IOC‘s
6.000	Controlled "devices"
5.000	Operator displays (~1500 unique)
18.000	Alarm-monitored PVs (!)
270.000	PVs



OTHER DEPLOYMENT TOOLS USED

Standard EPICS-base and -extensions runtime

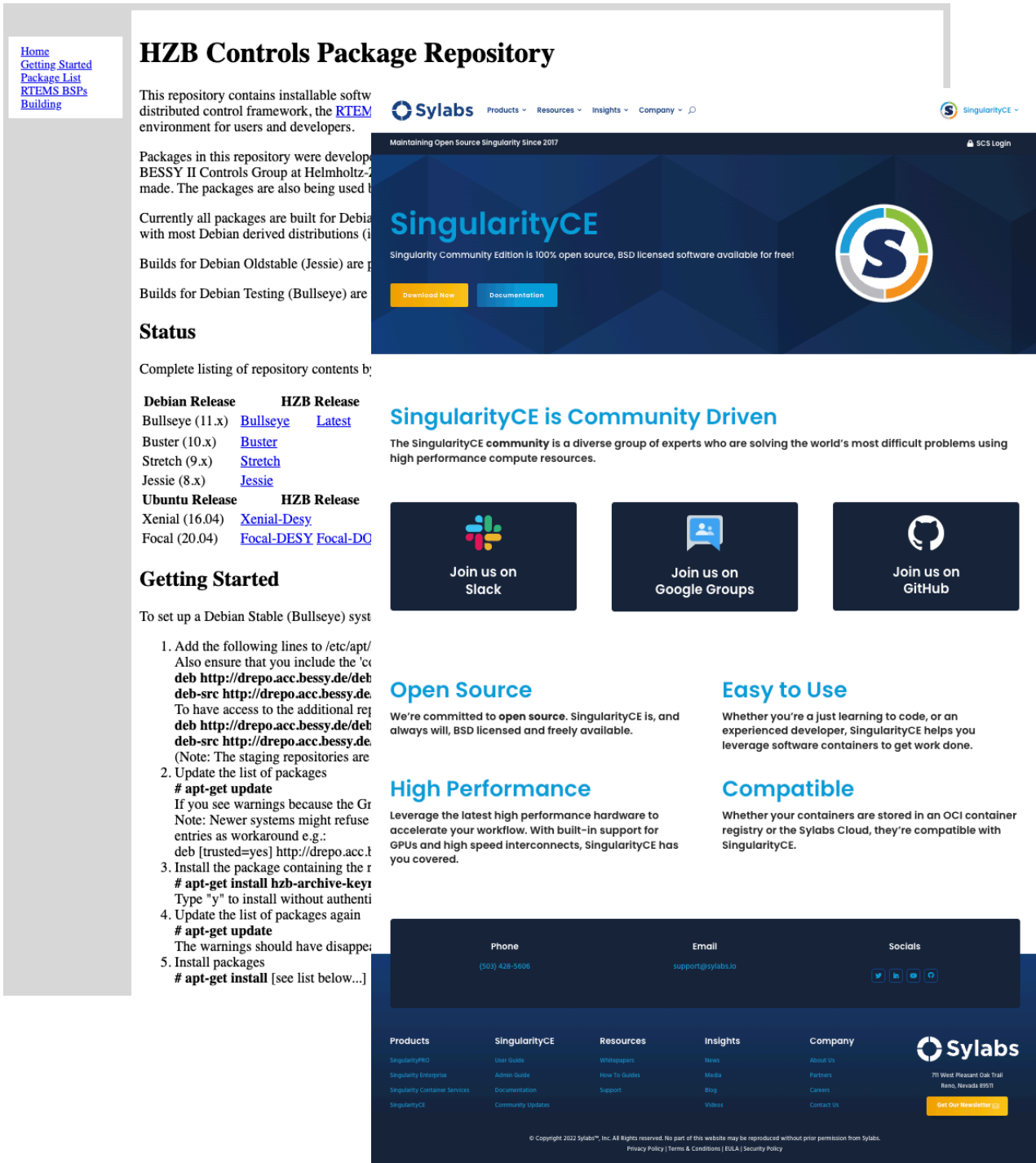
➡ ***Debian pkgs - local build- and repo-servers***

Legacy Software - renewal in progress, but...

- Old 32 bit binaries built >10 years ago
  neither 64 bit proof nor compiling today at all...
- Stable python environments including special stuff
  - ML tools, elegant/pelegant, octave, R,
    Tango-EPICS bridge, ...

➡ ***Singularity - https://sylabs.io/singularity/***

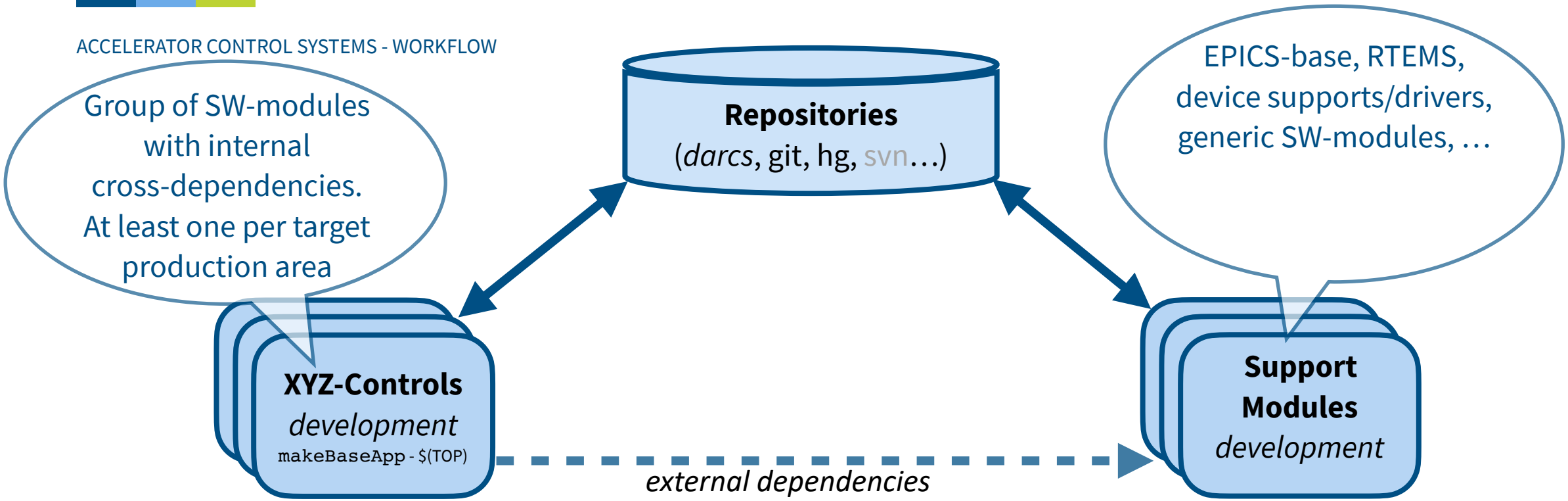
- perfect container system for standalone user level applications (my opinion!)



1. sumo

Build

see also full presentation by B. Franksen at EPICS Meeting 2019 @ ITER

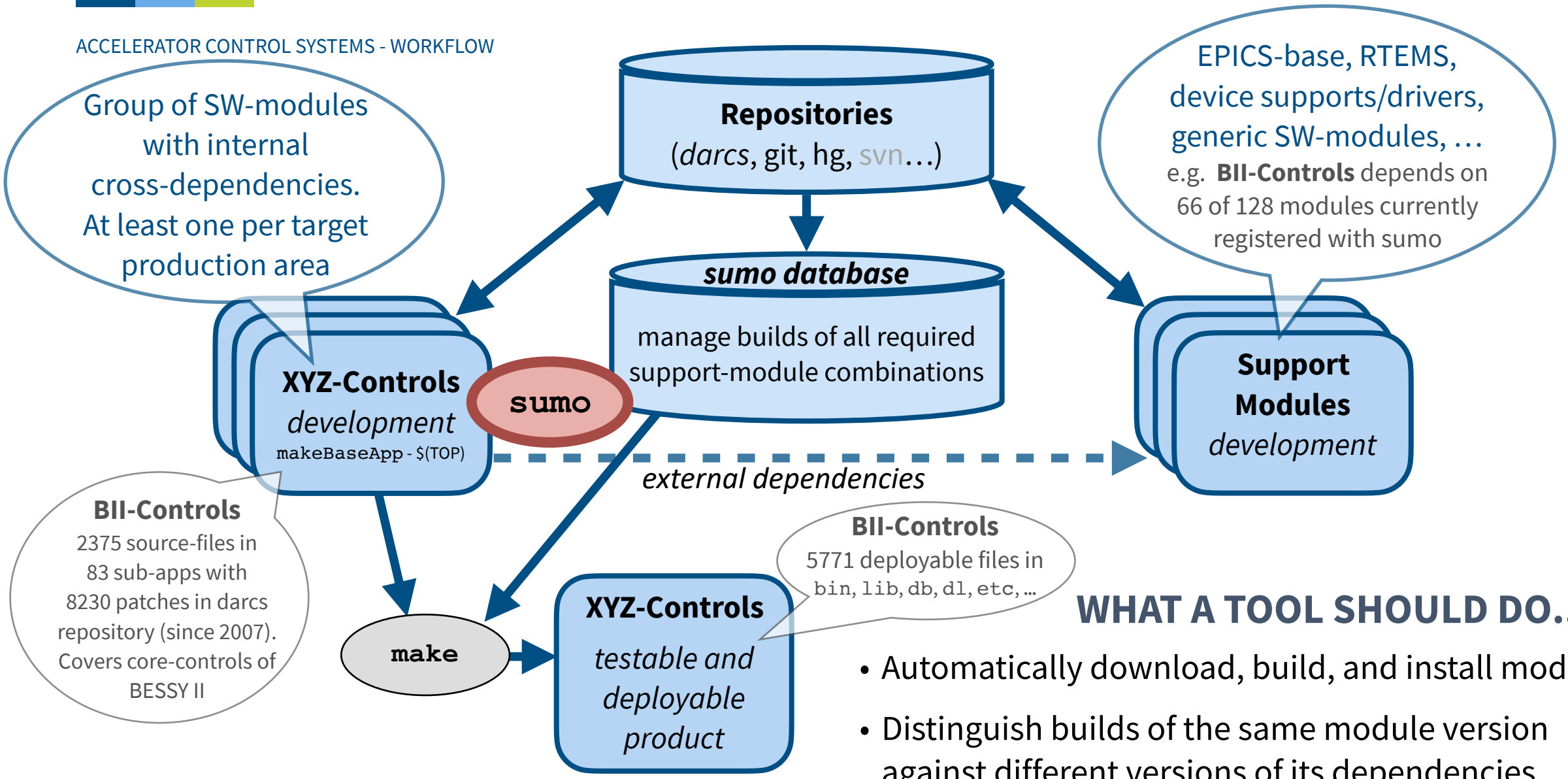


SUPPOSE...

- ... your application depends on a lot of support modules (base, device support, generic SW modules...)
  - Other applications depend on the same support modules - probably different versions (`{BII,BL,MLS,ERL,HoBiCaT,...}-Controls`)
  - Cross-dependencies between support modules
- ... something deep down in the dependency graph changes (base, asyn, sequencer,...)
- You need to re-build and re-install all dependent modules

SUPPOSE...

- ... your application depends on a lot of support modules (base, device support, generic SW modules...)
  - Other applications depend on the same support modules - probably different versions
  - Cross-dependencies between support modules (`{BII,BL,MLS,ERL,HoBiCaT,...}-Controls`)
- ... something deep down in the dependency graph changes (base, asyn, sequencer,...)
- You 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*)
  - Avoid that anything is forgotten to be rebuilt (*completeness*)
  - Invent a unique name (path) for the new builds so we don't break existing apps (*persistence*)



WHAT A TOOL SHOULD DO…

- Automatically download, build, and install modules
- Distinguish builds of the same module version against different versions of its dependencies
- Operate non-destructively: existing builds remain available for use by other applications

EXAMPLE OF A `configure/MODULES` FILE

End-application defines versions of every support module it directly depends on!

- What we need for that to work
  - Replace application's `configure/RELEASE` with something more abstract that declares module versions (`configure/MODULES`)
  - An external database of module versions with information about dependencies and source locations

```
{
  ...
  "module": [
    "ASYN:R4-35-bessy2",
    "AUTOSAVE:R5-10",
    "BASE:R3-15-6-bessy8",
    "DEVIOCSTATS:R3-1-9-bessy5",
    "MOTOR:R7-1",
    "SEQ:R2-2-9",
    ...
  ]
}
```

BUILDING SUPPORT MODULES FOR AN APPLICATION

```
> sumo build new
creating build with tag 'BII-185'
[...residual build output...]
```

- creates a unique fresh build identifier (here `BII-185`)
- 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*

SOURCE DEFINITION

```
> sumo build use
using build BII-185
```

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

```
> cat configure/RELEASE
# generated by sumo using build BII-185:
SEQ=/opt/Epics/sumo/build/SEQ/R2-2-9+BII-099
MOTOR=/opt/Epics/sumo/build/MOTOR/R7-1+BII-085
DEVIOCSTATS=/opt/Epics/sumo/build/DEVIOCSTATS/R3-1-9-bessy5+BII-085
AUTOSAVE=/opt/Epics/sumo/build/AUTOSAVE/R5-10+BII-085
ASYN=/opt/Epics/sumo/build/ASYN/R4-35-bessy2+BII-085
EPICS_BASE=/opt/Epics/sumo/build/BASE/R3-15-6-bessy8+BII-084
...
```

STATUS

- we are using it continuously since 2014
- works reliably, is actively maintained and stable
- author and maintainer is Götz Pfeiffer `goetz.pfeiffer@helmholtz-berlin.de`
- extensively documented:
  https://epics-sumo.sourceforge.io/
- written in Python3 (>=3.2)
- Mercurial repository
  `hg clone http://hg.code.sf.net/p/epics-sumo/mercurial epics-sumo`
- tar files, RPMs, and Debian packages:
  https://sourceforge.net/projects/epics-sumo/files/
- ... or just say
  `> pip3 install EPICS-sumo`

2. rsync-dist

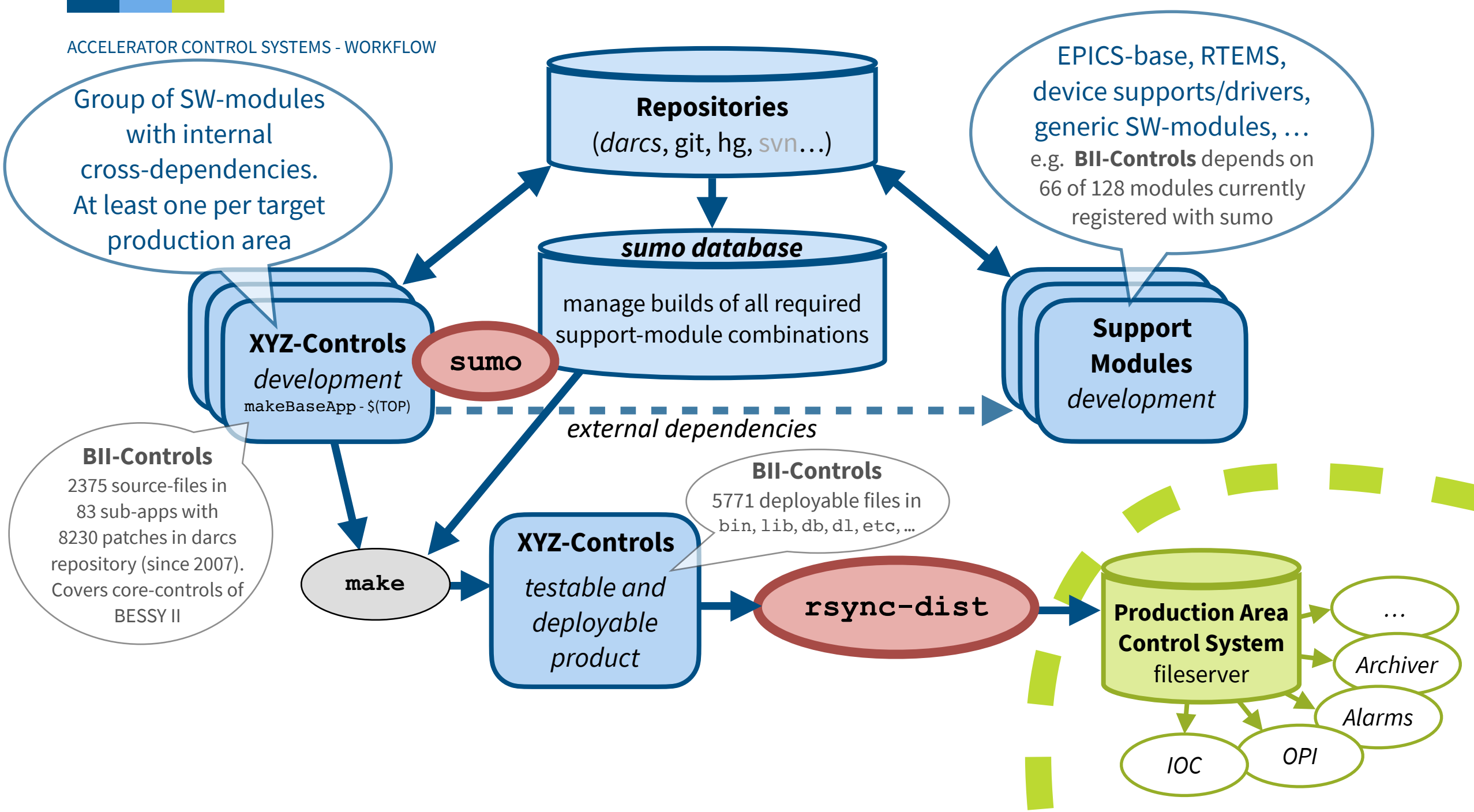
Deployment

see also full presentation by B. Franksen at EPICS Meeting 2019 @ ITER

TASK: DEPLOY EPICS APPLICATION'S BUILD RESULTS TO REMOTE SERVERS

Required features:

- deploy only what's needed (configurable, e.g. `db`, `dbd`, `bin`, `dl`, `etc`, ...)
- allow incremental update (i.e. one IOC at a time)
- easy and quick fallback to previous versions
- don't waste too much disk space
- conveniently install test and debug versions
  - scientific facilities often contain unique / prototype devices
- light-weight usage
  - don't require developers to invent names or log messages
  - no (explicit) login, execute from the command line inside developer's source tree


Group of SW-modules with internal cross-dependencies. At least one per target production area
Repositories
(darcs, git, hg, svn…)
EPICS-base, RTEMS, device supports/drivers, generic SW-modules, …
e.g. BII-Controls depends on 66 of 128 modules currently registered with sumo
sumo database
manage builds of all required support-module combinations
XYZ-Controls
development
makeBaseApp - $(TOP)
sumo
Support Modules
development
external dependencies
BII-Controls
2375 source-files in 83 sub-apps with 8230 patches in darcs repository (since 2007). Covers core-controls of BESSY II
BII-Controls
5771 deployable files in bin, lib, db, dl, etc, …
make
XYZ-Controls
testable and deployable product
rsync-dist
Production Area Control System
fileserver
…
Archiver
Alarms
IOC
OPI

A MINIMALISTIC APPROACH

- the basic idea
  - deployed version = *directory*
  - version name = *timestamp*
  - symbolic name = *symlink to version*
- log files on server side and locally in user's home for auditing
- use **`ssh`** and **`ssh-agent`** for password-less authorisation and authentication
- reduce disk space overhead by sharing identical files over releases
  - create a recursive hard-linked copy of the version last distributed by this user via **`ssh`**
  - then pull new build artefacts using **`rsync -a --delete`** from the developer's directory
  - **`rsync`** does the right thing here:
    - existing identical files are left alone
    - otherwise old version is removed before copying

```
...
-rw-r--r--   3 opiadm opiadm    2430 Sep 13 10:16 TAI.arch
-rw-r--r-- 158 opiadm opiadm   10910 Sep 13 10:16 TopUp.arch
-rw-r--r-- 531 opiadm opiadm   34641 Sep 13 10:16 TopUpEffIlk.arch
-rw-r--r-- 273 opiadm opiadm     385 Sep 13 10:16 TRIBsCurrents.arch
-rw-r--r--  14 opiadm opiadm   21983 Sep 13 10:16 TriggerDelay.arch
-rw-r--r-- 781 opiadm opiadm    3776 Sep 13 10:16 TuneFDBK.arch
...
```

COMMANDS

> **`rsync-dist.pl -c CONFIGFILE dist`**
  - create a unique new version (directory) in the form of a time stamp e.g. **`2021-09-12T15:38:29`**

> **`rsync-dist.pl -c CONFIGFILE change-links -L NAME`**
  - change the symbolic link **`NAME`** to the last version distributed by this user
  - used to *activate* a new version for some IOC

> **`rsync-dist.pl -c CONFIGFILE change-links VERSION,NAME`**
  - change the symbolic link **`NAME`** to the given **`VERSION`**
  - used for *fallback* to a previous version

Completely integrated into **`Makefile`**-system - few extra rules:

```
> make rsync-dist.OPI
> make activate-version.OPI NAMES=PROD
> make rsync-dist.IOC activate-version.IOC NAMES=IOC2S12G,SIOC20C
```

EXAMPLE OF A CONFIGURATION FILE

```
> cat configure/rsync-dist.IOC.config
RSYNC_DIST_HOST=iocadm@nfs.ctl.bessy.de
RSYNC_DIST_PATH=/opt/IOC/BII-Controls/dist
RSYNC_DIST_LINKPATH=/opt/IOC/BII-Controls/links
RSYNC_DIST_PREFIX_DISTDIR=1
RSYNC_DIST_LOCALPATH=bin,db,dbd,iocBoot
RSYNC_DIST_CHECKSUM=1
```

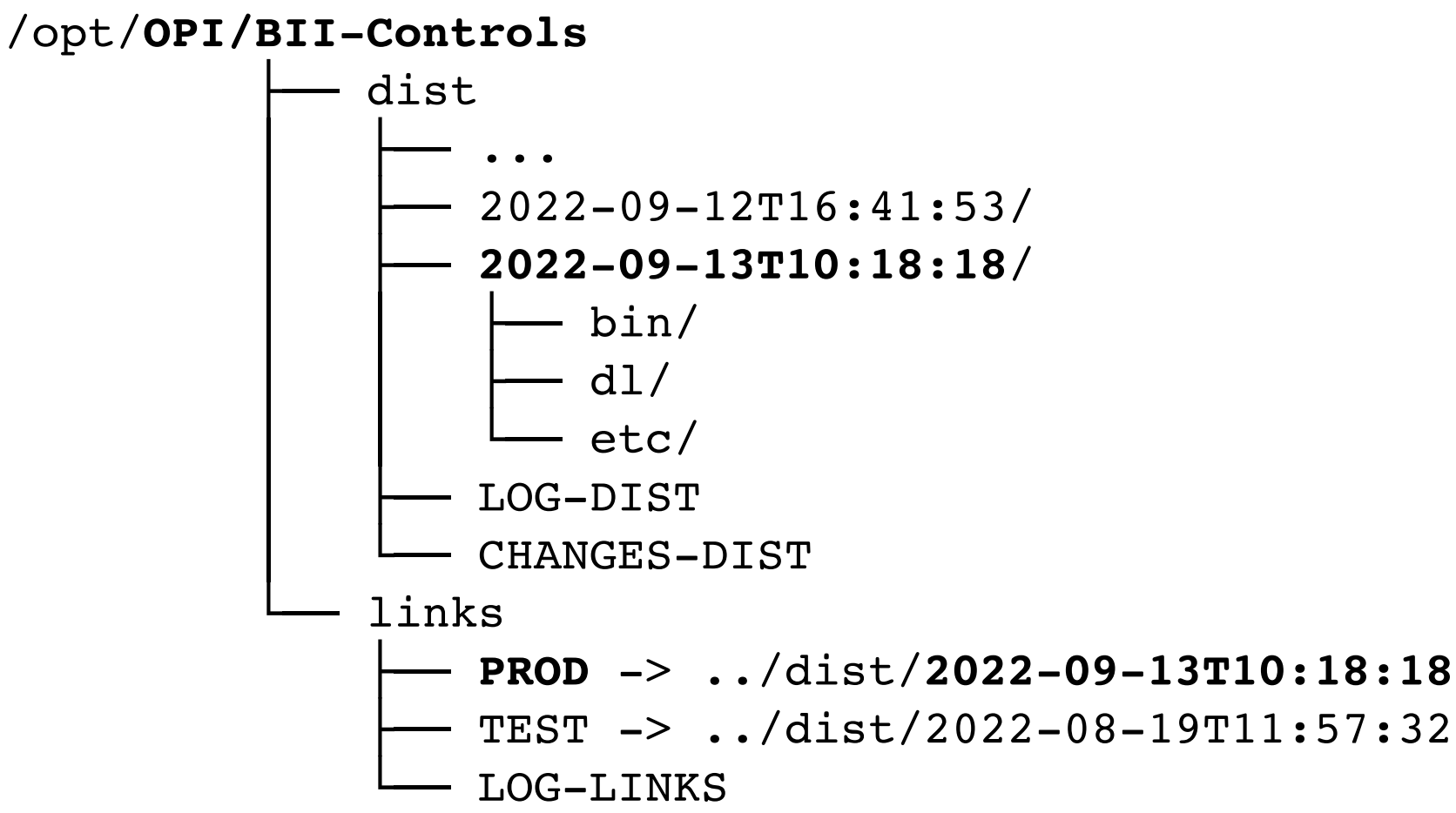
Remarks:

- Configuration file usually kept in application's `configure` directory
- Many more configuration options available

STRUCTURE ON DEPLOYMENT-TARGET - IOC RELATED

```
/opt/IOC/BII-Controls
├── dist
│   ├── ...
│   ├── 2022-09-12T13:15:16/
│   └── 2022-09-12T15:38:29/
│       ├── bin/
│       ├── db/
│       ├── dbd/
│       ├── iocBoot/
│       └── ...
└── links
    ├── IOCS10G -> ../dist/2022-09-05T09:59:20
    ├── SIOC19C -> ../dist/2021-07-30T12:46:23
    ├── SIOC20C -> ../dist/2022-09-12T15:38:29
    └── ...
```

STRUCTURE ON DEPLOYMENT-TARGET - OPI RELATED



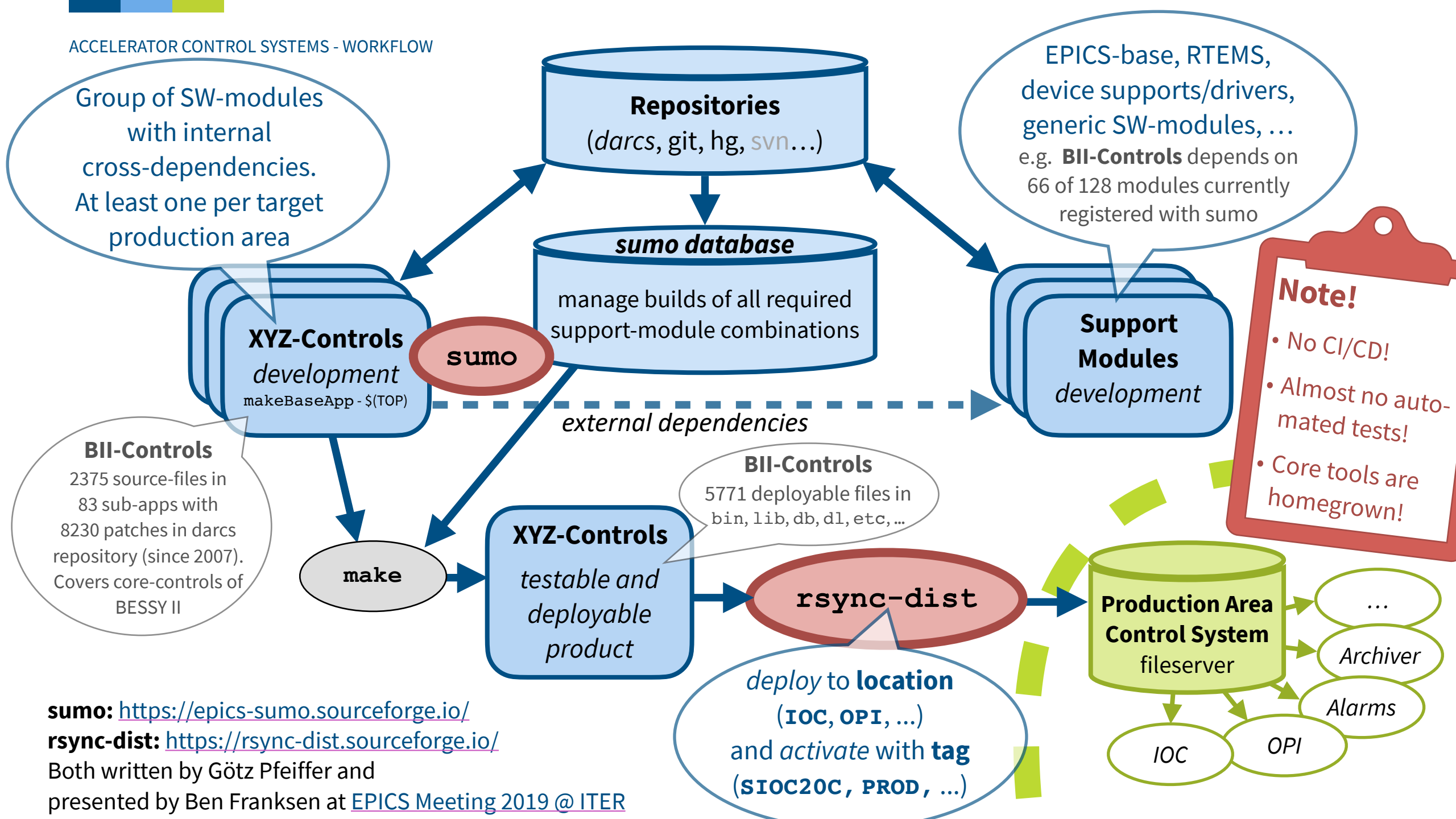
STATUS

- in production use since 2007
- implemented in Perl
- pretty old code with a number of quirks, such as
  - does not like being interrupted (leaves lock files)
  - CLI could be more user-friendly, too many options
  - version/timestamp contains colon, tends to confuse third-party tools like ssh
- could use a clean re-write from scratch
- but the basic idea is sound and stood the test of time

Docs: https://rsync-dist.sourceforge.io/

Source: `darcs clone https://www-csr.bessy.de/control/bii_scripts/repo/bii_scripts/`

Questions: `goetz.pfeiffer@helmholtz-berlin.de`



sumo: https://epics-sumo.sourceforge.io/
rsync-dist: https://rsync-dist.sourceforge.io/
Both written by Götz Pfeiffer and presented by Ben Franksen at EPICS Meeting 2019 @ ITER

Thanks a lot for listening!

Questions? Remarks?