

Concept and Implementation of an Efficient Version Control System for SAS® Programs in Clinical Development

**PhUSE Annual Conference
12th October 2015**

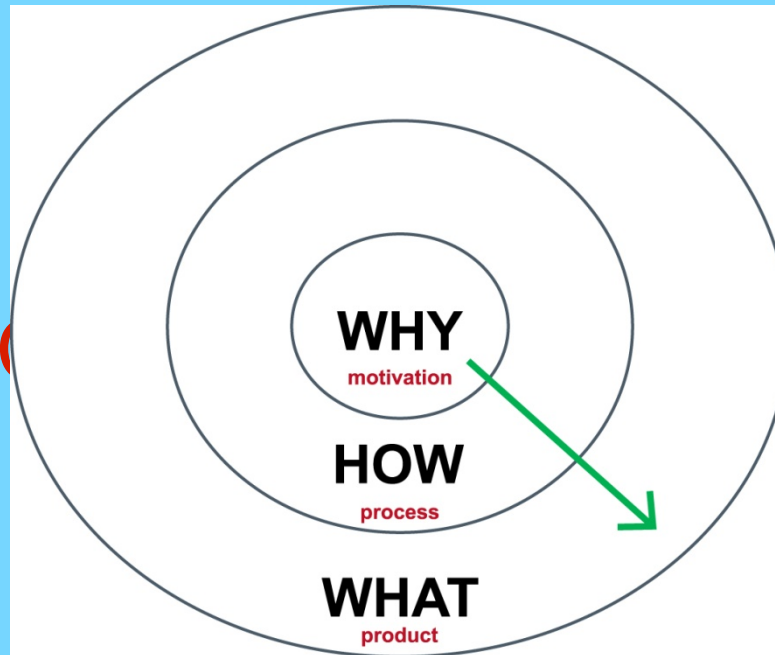
Matthias Post, Accovion, Eschborn, Germany

Dirk Spruck, Accovion, Marburg, Germany

Agenda

- Motivation

-



- Conclusion

Agenda

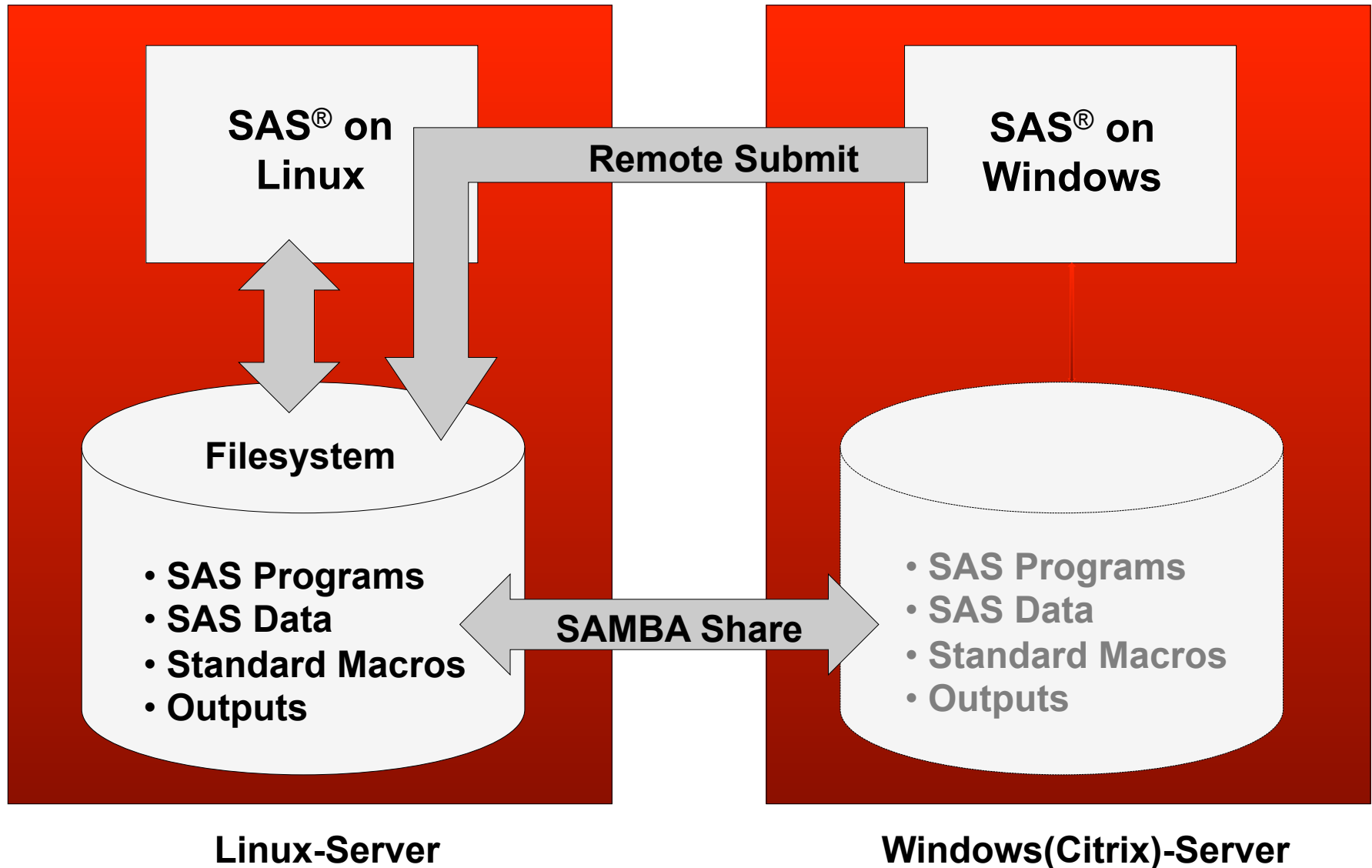
- Motivation
- Accovion's SAS[®] programming environment
- Program development & validation concept
- The Version Control System
 - Functions
 - Archiving of programs
 - The history database
- Conclusion

Why use a Version Control System ?

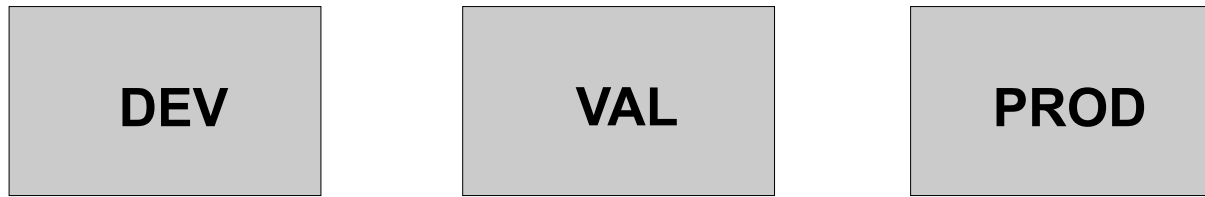
- Usual requirements in program development
 - Recover earlier program versions
 - Compare program versions
 - Create backup copies, e.g. before major changes
- Needs of regulatory authorities
 - Code validated
 - Results reproducible
 - Programs retrievable

No built-in version control for code in SAS environment

SAS®- Architecture at Accovion



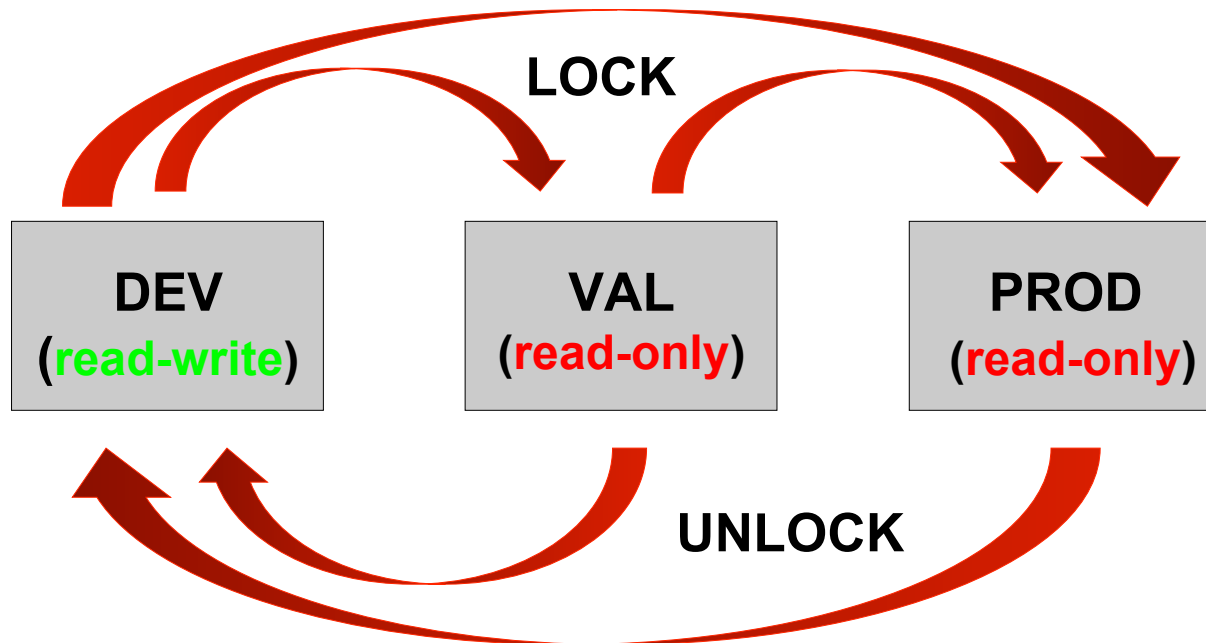
Development / Validation levels for SAS® programs



Directories reflect the status of a program

- ✓ DEV : programs in development phase
- ✓ VAL : completed programs that need to be validated
- ✓ PROD : validated programs

The Locking Process for SAS® programs

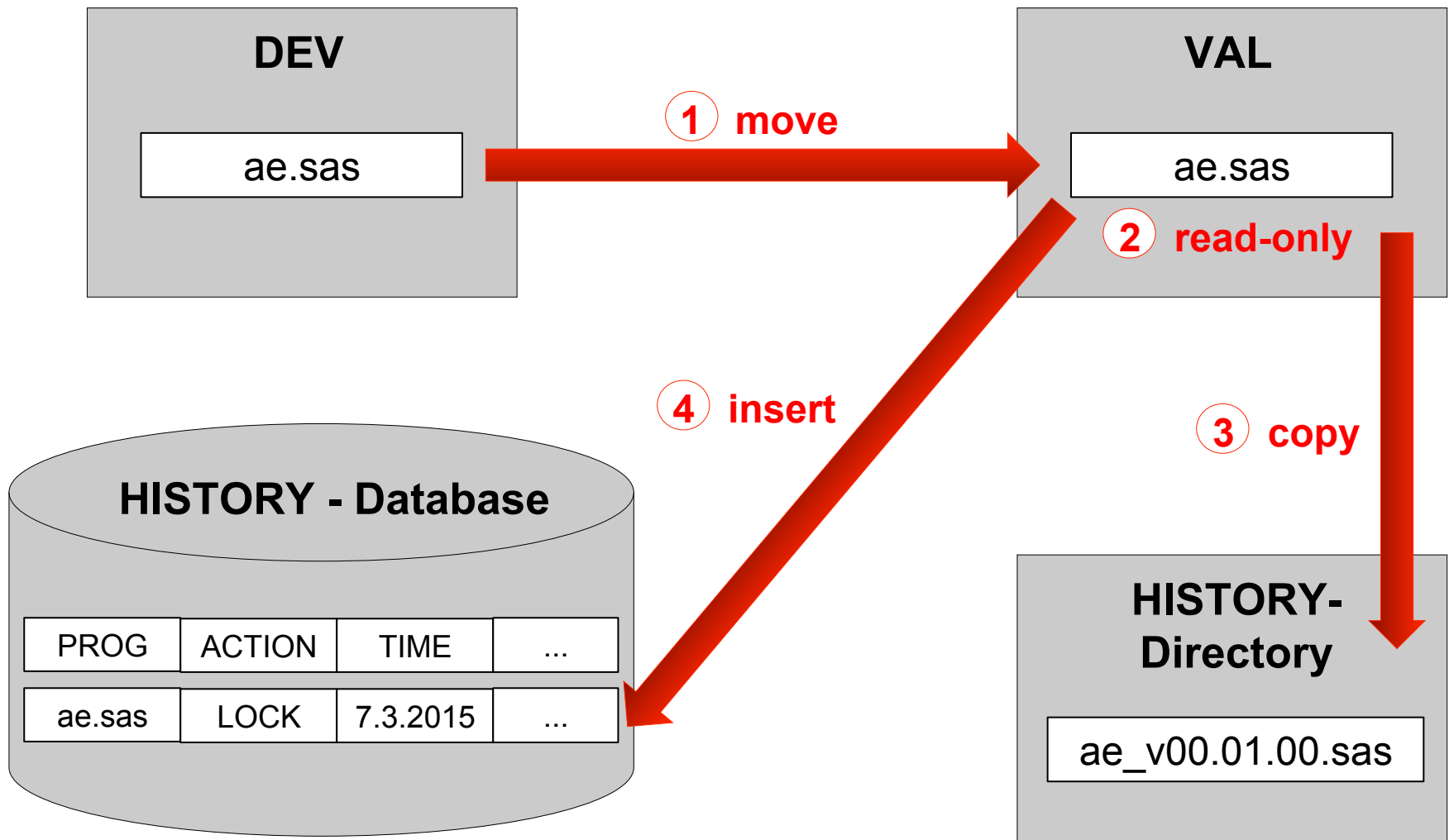


- ✓ **LOCK :** Move of a program to the next higher validation level. Permissions are set to "**read-only**"
- ✓ **UNLOCK:** Move of a program back to DEV. Permissions are set back to "**read-write**"

The Version Control System

- Set of functions implementing the lifecycle of SAS programs:
 - moving programs between validation levels (DEVELOPMENT, VALIDATION, PRODUCTION)
 - managing access rights (read-only for VAL+ PROD)
 - storing backup copies & assigning version numbers

Functionality of the versioning tools (Example : lock)



Syntax example : The LOCK function

Moves one or more files “up” one level (DEV -> VAL, VAL-> PROD)

```
lock [-l] [-c "comment"] file1 [file2 ...]
```

- c adds a comment to the corresponding record in the versioning database
- l For related .log and .lst files a backup copy with the same version number is created.
("related" := same filename + same directory)
- Use of wildcards to process multiple files
- Can be used also to version other file types (e.g. related Excel or csv files)

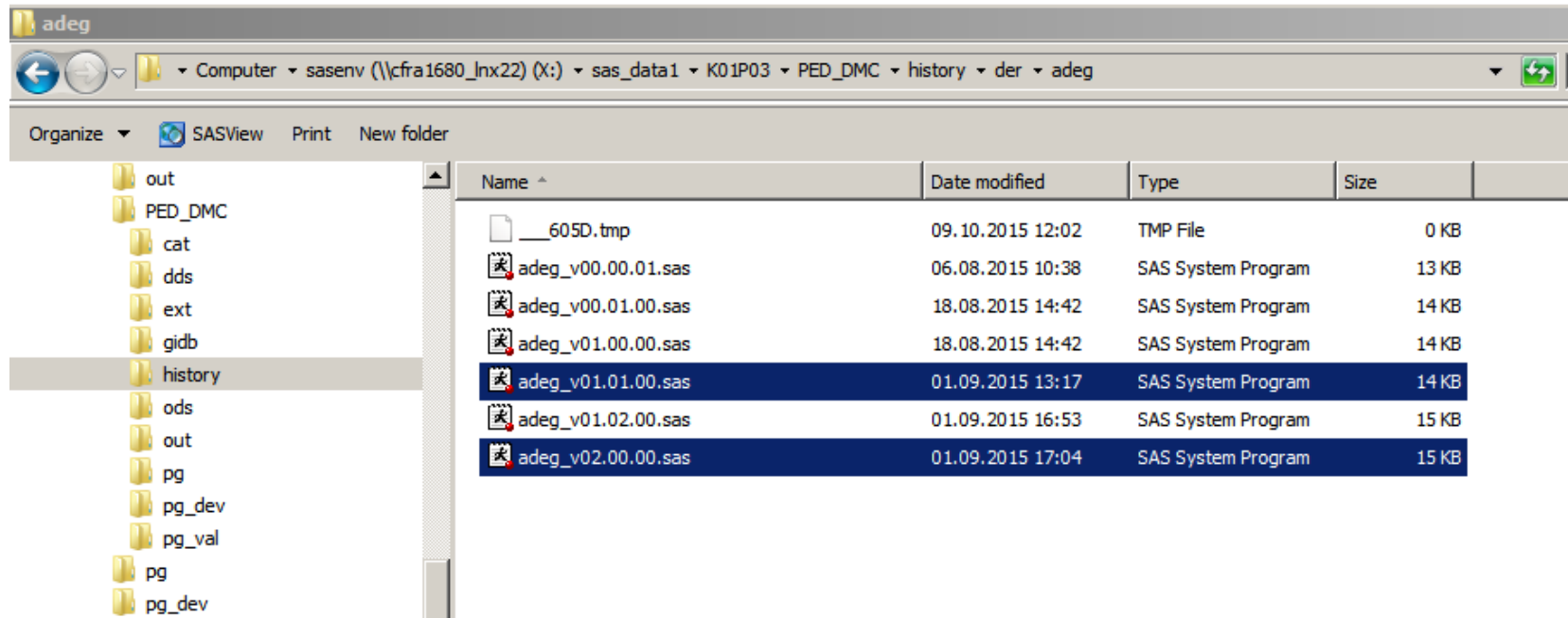
Additional functions

- **UNLOCK**
Moves a program back to DEV level and sets write permissions
- **RENAME**
"controlled" rename of a program.
Needed because backup copies and database entries are identified by program name and path
- **MODIFY**
Take over ownership of a program (which is necessary to edit/modify it)
Only possible for other study team members

An email notification is sent to the original owner.
- **SAVE**
Creates an intermediate backup copy during development phase

Usage example : Comparing two program versions

- Step 1: On the Windows/Citrix-Server, identify and mark two program versions :



- Step 2:
Start „SourceGear DiffMerge“ by a rightclick to compare

The screenshot shows the SourceGear DiffMerge application window titled "adeg_v01.01.00.sas, adeg_v02.00.00.sas - SourceGear DiffMerge". The window displays a side-by-side comparison of two SAS programs. The left pane shows the original file "X:\sas_data1\K01P03\PED_DMC\history\der\adeg\adeg_v01.01.00.sas" and the right pane shows the modified file "X:\sas_data1\K01P03\PED_DMC\history\der\adeg\adeg_v02.00.00.sas". The code in both panes is identical except for line 134 in the right pane, where the variable list is updated to include "aage aageu age_ve tapentfl". The interface includes a menu bar (File, Edit, View, Tools, Help), a toolbar with icons for file operations, and a status bar at the bottom showing "Reference View (Files as Loaded)", "Edit View (File as Edited)", "Ruleset: _Default_", and "Changes: 4".

```

130 run;
131
132 **** Merge with ADSL to get Age information ***;
133
134 proc sort data=dds.adsl (keep=studyid usubjid age ageu)
135     out=adsl0;
136     by studyid usubjid ;
137 run;
138
139 data eg2;
140     merge eg1 (in=in_eg)
141           adsl0 (in=in_adsl);
142     by studyid usubjid ;
143     if in_eg;
144     if not in_adsl then do;
145         * put "WAR" "NING : patient not found in adsl :" studyid= usubjid=;
146         delete;
147     end;
148
149 *** Derive help variables for age ***;
  
```

```

130 run;
131
132 **** Merge with ADSL to get Age information ***;
133
134 proc sort data=dds.adsl (keep=studyid usubjid age ageu aage aageu age_ve tapentfl)
135     out=adsl0;
136     by studyid usubjid ;
137 run;
138
139 data eg2 (drop=tapentfl);
140     merge eg1 (in=in_eg)
141           adsl0 (in=in_adsl);
142     by studyid usubjid ;
143     if in_eg;
144     if not in_adsl then do;
145         * put "WAR" "NING : patient not found in adsl :" studyid= usubjid=;
146         delete;
147     end;
148
149 *** Derive help variables for age ***;
  
```

The History Database

- All actions of any versioning function are tracked in an ORACLE table.
- Each action is creating exactly one new record
- Read-access via SAS ACCESS TO ORACLE
- Possible use :
 - Show the lifecycle of a specific program
 - Get a quick overview on the status of a study :
Which programs are under development, which are productive?

PROJECT	STUDY	PATH	FILENAME	ACTION	TIMESTAMP	USERID	SOURCE	TARGET	ARCFN
K99P09	XYZ_PHASE_III	rep/eff	fg01.sas	MOD	18NOV2013:08:55:53	cfra1827	pg_dev	pg_dev	fg01_v0
K99P09	XYZ_PHASE_III	rep/eff	fg01.sas	LOCK	19NOV2013:14:29:41	cfra1827	pg_dev	pg_val	fg01_v0
K99P09	XYZ_PHASE_III	rep/eff	fg01.sas	LOCK	19NOV2013:14:37:43	cfra1827	pg_val	pg	fg01_v0
K99P09	XYZ_PHASE_III	rep/consultancy	g014.sas	REN	20NOV2013:15:09:21	cfra1124	pg_dev	pg_dev	g014_v
K99P09	XYZ_PHASE_III	rep/eff	fg14.sas	UNLOCK	20NOV2013:15:13:56	cfra1124	pg_val	pg_dev	
K99P09	XYZ_PHASE_III	rep/consultancy	fg14.sas	REN	20NOV2013:15:16:26	cfra1124	pg_dev	pg_dev	fg14_v0
K99P09	XYZ_PHASE_III	rep/consultancy	fg14.sas	LOCK	20NOV2013:15:19:51	cfra1124	pg_dev	pg_val	fg14_v0
K99P09	XYZ_PHASE_III	rep/consultancy	g014.sas	LOCK	20NOV2013:15:23:59	cfra1124	pg_dev	pg_val	g014_v
K99P09	XYZ_PHASE_III	rep/consultancy	ae03c.sas	LOCK	20NOV2013:15:29:11	cfra1124	pg_dev	pg_val	ae03c
K99P09	XYZ_PHASE_III	rep/consultancy	ae03c.sas	LOCK	20NOV2013:15:34:35	cfra1124	pg_val	pg	ae03c
K99P09	XYZ_PHASE_III	rep/consultancy	fg14.sas	LOCK	21NOV2013:11:07:39	cfra1124	pg_val	pg	fg14_v0
K99P09	XYZ_PHASE_III	rep/consultancy	g014.sas	LOCK	21NOV2013:11:08:47	cfra1124	pg_val	pg	g014_v
K99P09	XYZ_PHASE_III	rep/rmp2014	rmpaes.sas	UNLOCK	21NOV2013:11:12:22	cfra1827	pg	pg_dev	
K99P09	XYZ_PHASE_III	der/302	advs302.sas	UNLOCK	21NOV2013:15:29:18	cfra1124	pg	pg_dev	



Example :
Contents of the
history database



GET	ARCFNAME	PREVUSERID	PREVNAME	COMMENT
lev	fg01_v00.02.03.sas	cfra1124		
al	fg01_v00.03.00.sas			
	fg01_v01.00.00.sas			
lev	g014_v00.01.01.sas		rep/eff/g014.sas	
lev				locked by mistake
lev	fg14_v00.01.01.sas		rep/eff/fg14.sas	
al	fg14_v00.02.00.sas			
al	g014_v00.02.00.sas			
al	ae03c_v00.01.00.sas			
	ae03c_v01.00.00.sas			
	fg14_v01.00.00.sas			
	g014_v01.00.00.sas			validated by x.y.
lev				
lev		cfra9173		

Technical Details

- System is based on Linux-Shell-Scripts
- Approx. 1500 lines of program code (without comments)
- Creation of database entries :
SQL-Insert-Statements are generated by the Linux scripts
and are sent via SQLPLUS to the database.
- DB entries are cached in a folder if DB is offline
Can be re-loaded at a later time
=> higher availability
- SAS-Code is only used for accessing the database

Conclusion

- Advantages :
 - Low development effort (done inhouse)
 - Built on existing software components
 - Independent from release changes of the underlying components (LINUX, SAS®, ORACLE®)
 - Slim, transparent & easy to use
 - Highly appreciated by the users (clear benefit)
- Drawbacks :
 - Could be bypassed by users
(e.g. manual moving / renaming of programs)
=> good training and awareness are essential

Questions ?

ACCOVION

Dirk Spruck

Principal Statistical
Programmer

www.accovion.com

Accovion GmbH
Software Center 3
D-35037 Marburg,
Germany

Tel. +49 6421 94849-37
dirk.spruck@accovion.com

ACCOVION

Matthias Post

Statistical Programmer

Accovion GmbH
Helfmann-Park 10
D-65760 Eschborn, Germany
Tel. +49 6196 7709-274
matthias.post@accovion.com

www.accovion.com

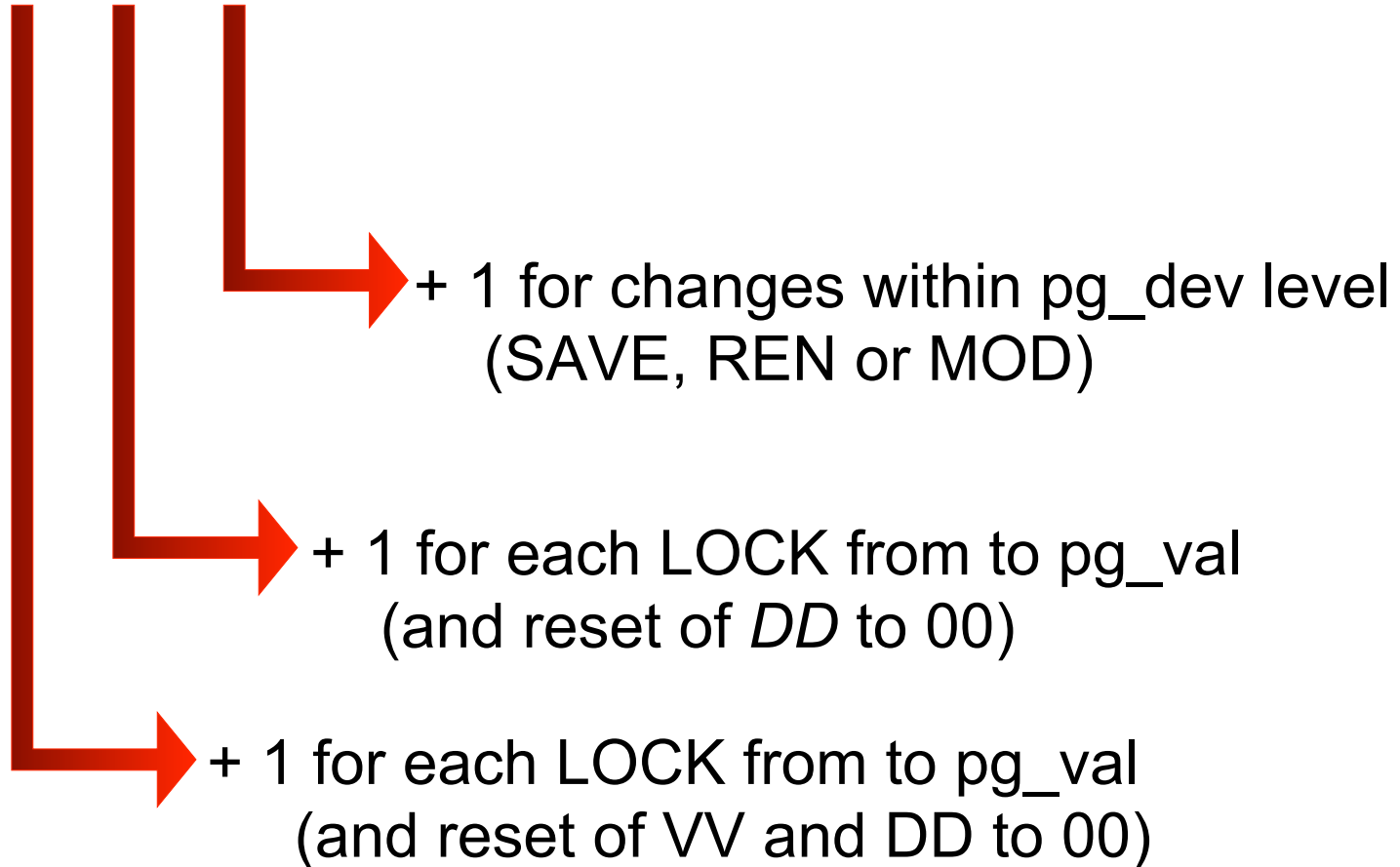
Backup slides

Version numbering

- Example : ecg.sas => ecg_v00.02.01.sas
- 3 level version number : PP.VV.DD
- Lowest Level ("DEV level", rightmost)
 - Will be increased for all operations within pg_dev (e.g. MOD, REN, SAVE)
- Mid Level ("VAL level")
 - Will be increased in case of a LOCK from pg_dev to pg_val
 - At this time the "DEV level" version number will be reset to 00
- Highest Level ("PROD level", leftmost)
 - Will be increased in case of a LOCK from pg_dev or pg_val to pg
 - At this time all lower level version numbers will be reset to 00

Version numbers

<filename>_vPP.VV.DD.sas



Contents of the History Database

- Key attributes : project, study, program name & path
- Action (LOCK, UNLOCK, RENAME, MODIFY, SAVE)
- Validation levels (before action / after action)
- Name and path of backup copy, version number
- Related log-/listfiles (if archived together with program)
- Username and time of the action
- Last modification date of the program
- Comment (specified by the -c option)
- In case of a RENAME : previous program name and path
- In case of a MODIFY : previous program owner

What we are doing at Accovion

- CRO : Contract Research Organization
Planning, conduct, and analysis of clinical studies
- SAS® Programs are used for several purposes, e.g. for the generation of
 - Standardized datasets (CDISC/SDTM + ADAM)
 - Tables & listings
 - Figures & graphs