Practical experiences from the Munich NGS-FabLab – Tools, compatibility and pitfalls off the standard track

Aarif Mohamed Nazeer Batcha[2], Sebastian Schaaf [1,2], Guokun Zhang[2], Sandra Fischer[2], Ashok Varadharajan[2], Ulrich Mansmann[1,2]

[1] German Cancer Consortium (DKTK), Heidelberg, Germany
[2] Department of Medical Informatics, Biometry and Epidemiology (IBE), Ludwig Maximilians University (LMU) Munich, Germany

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

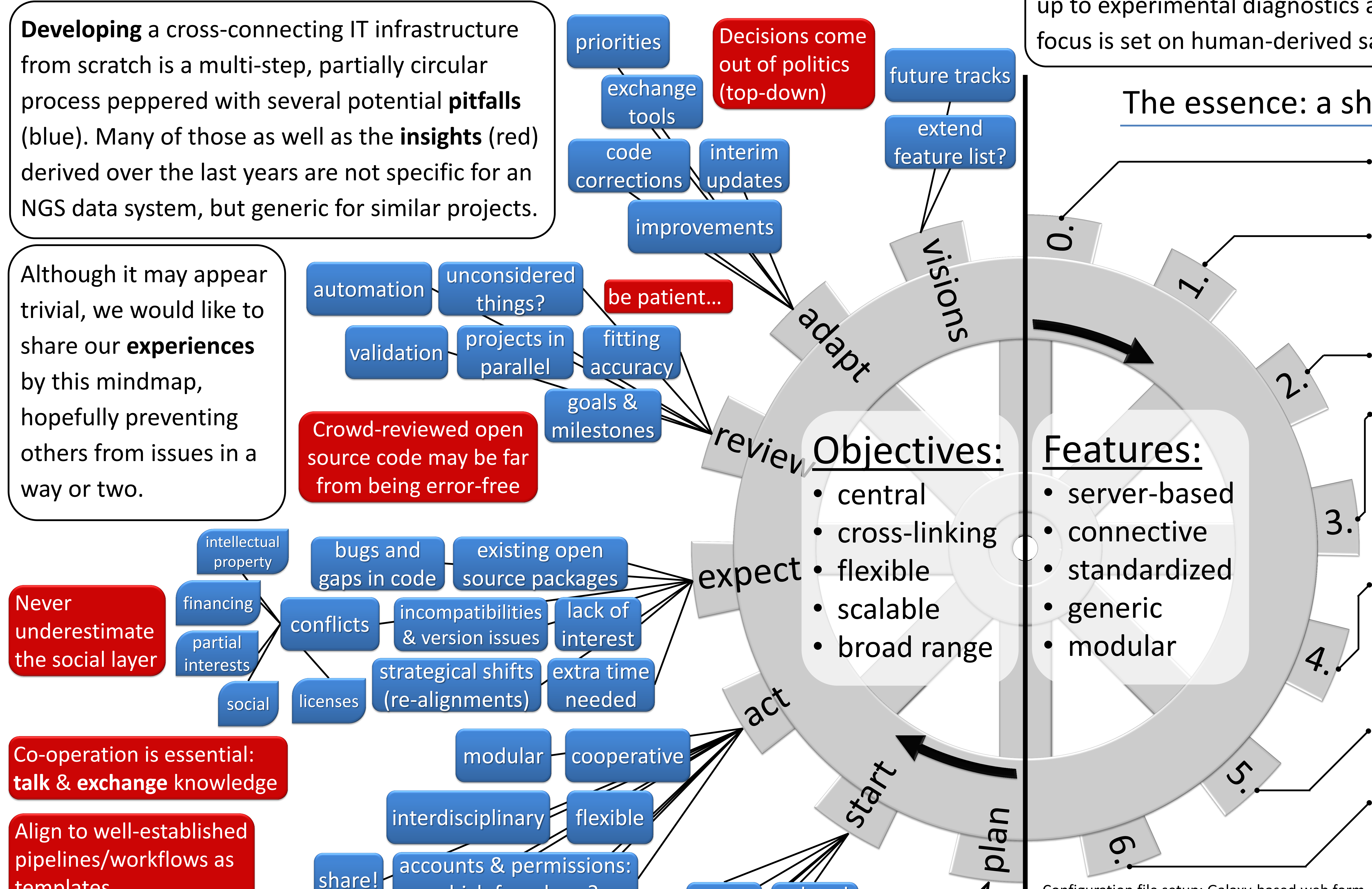
MEDIZINISCHE FAKULTÄT
INSTITUT FÜR MEDIZINISCHE INFORMATIONSVERARBEITUNG
BIOMETRIE UND EPIDEMIOLOGIE

DEUTSCHES KREBSFORSCHUNGSZENTRUM IN DER HELMHOLTZ-GEMEINSCHAFT

The NGS-FabLab is the first multi-institutional IT infrastructure for research at the university hospital and medical faculty of the LMU (which are in fact two juristic persons) in Munich, supporting molecular biomedical and clinical researchers from basic research up to experimental diagnostics and translational studies. Also due to data security laws a focus is set on human-derived samples, but not restricted to (mice, microbes, …).

Developing a cross-connecting IT infrastructure from scratch is a multi-step, partially circular process peppered with several potential **pitfalls** (blue). Many of those as well as the **insights** (red) derived over the last years are not specific for an NGS data system, but generic for similar projects.

Although it may appear trivial, we would like to share our **experiences** by this mindmap, hopefully preventing others from issues in a way or two.



visions
future tracks
extend feature list?

adapt
priorities
exchange tools
code corrections
interim updates
improvements
Decisions come out of politics (top-down)

review
automation
unconsidered things?
be patient…
validation
projects in parallel
fitting accuracy
goals & milestones
Crowd-reviewed open source code may be far from being error-free

expect
intellectual property
bugs and gaps in code
existing open source packages
financing
conflicts
incompatibilities & version issues
lack of interest
partial interests
social
licenses
strategical shifts (re-alignments)
extra time needed
Never underestimate the social layer

act
modular
cooperative
interdisciplinary
flexible
Co-operation is essential: **talk** & **exchange** knowledge
Align to well-established pipelines/workflows as templates

start
share!
accounts & permissions: which for whom?
define alpha projects & partners
common knowledge base
include students!
exclude troublemakers
record in writing
who's in, who's out?
know local the constraints/policies
Integrate the local admins
Students are unbelievable invaluable staff members
Facts come from the working level (bottom-up)
Refer to as many standards as possible (do not try to reinvent the wheel)
Moving montains does not work as a one-man show

plan
survey
related projects
workshops & conferences
requirements and conditions
setting/ conditions
deciders & makers
organizational structure
menpower (3-5 positions min.)
financing

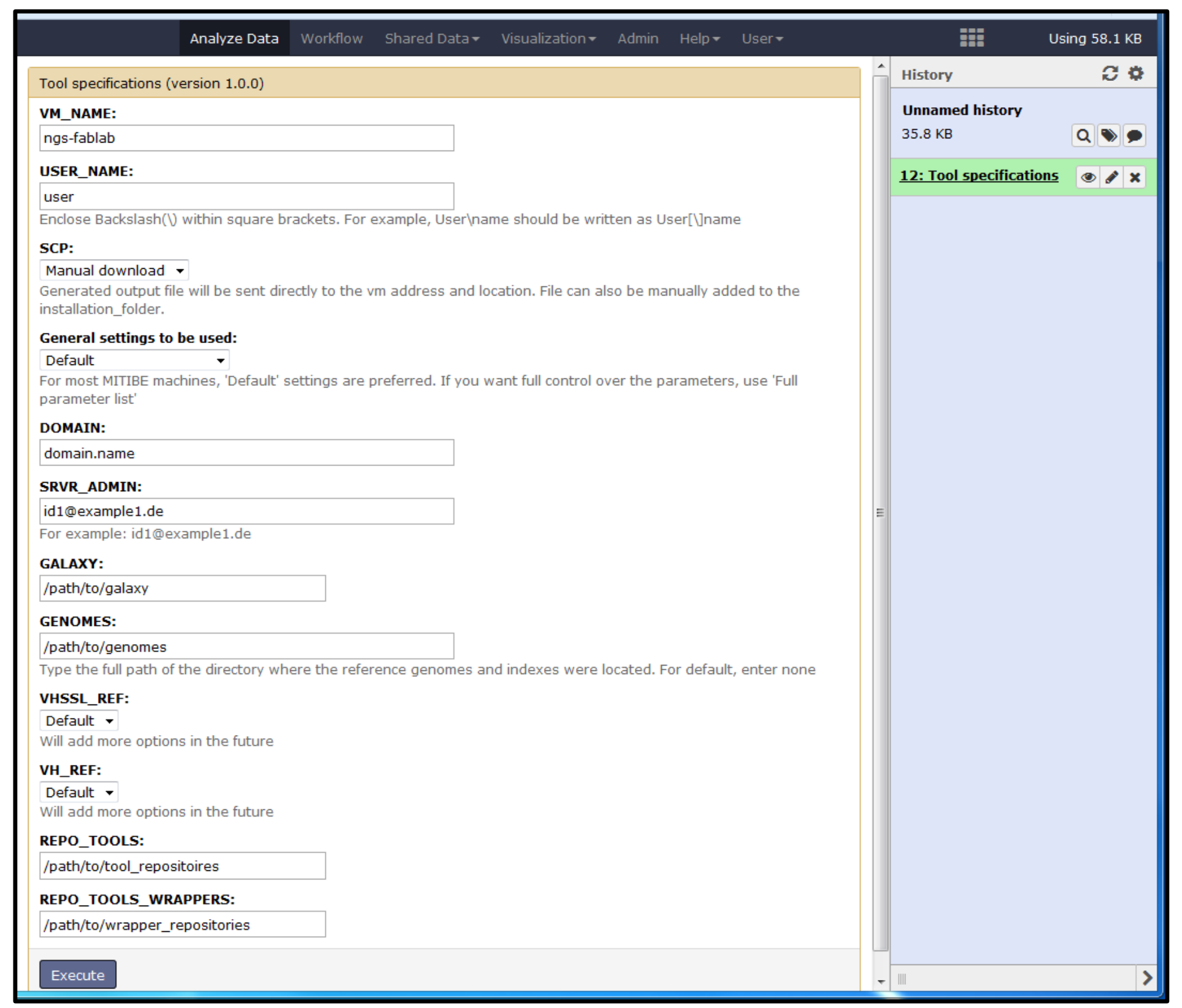
Objectives:
- central
- cross-linking
- flexible
- scalable
- broad range

Features:
- server-based
- connective
- standardized
- generic
- modular

The essence: a shell setup script

0. Create configuration file
   - editor-derived *or*
   - interactive from command line *or*
   - Galaxy web form
1. Preliminary inquiry
   - root user?
   - shell environment?
   - process user inputs
   - list tools to be installed
   - connect to repositories required for installation procedure
2. Connections setup (opt.)
   - NFS remote sites (storage, shared `tmp`)
   - SGE Cluster(s) (on going)
3. Galaxy Installation
   - set proxies
   - galaxy user setup, human users, permissions
   - get Galaxy and update (own repo or official `galaxy-dist`)
   - Backup important files and folders (`universe_wsgi.ini`, `tool_conf.xml`, `tool_data_table_conf.xml`, folders `tools` and `tool-data`)
   - Edit `universe_wsgi.ini` in accordance with the configuration file
4. Tools installation (home scheme; opt.)
   - Download, extract and install specified tools from our repository
   - Integrate tools' version control in Galaxy interface (on going)
   - Install libraries and supporting tools
   - Download and install tool wrappers from our repository
   - Edit `tool_conf.xml`, `tool_data_table_conf.xml`, tool specific `.loc` files and `.bashrc`
5. Reference genomes
   - currently `hg18` and `hg19` with tool-specific index files and dictionaries
   - microbial genomes in preparation
6. Further system-wide setups/settings (opt.)
   - Apache with configurations and customized webpage
   - PostgreSQL database and configurations
   - Active Directory connection in preparation

Configuration file setup: Galaxy-based web form



Tool specifications (version 1.0.0)

VM_NAME: ngs-fablab

USER_NAME: user
Enclose Backslash(\) within square brackets. For example, User\name should be written as User[\]name

SCP: Manual download
Generated output file will be sent directly to the vm address and location. File can also be manually added to the installation_folder.

General settings to be used: Default
For most MITIBE machines, 'Default' settings are preferred. If you want full control over the parameters, use 'Full parameter list'

DOMAIN: domain.name

SRVR_ADMIN: id1@example1.de
For example: id1@example1.de

GALAXY: /path/to/galaxy

GENOMES: /path/to/genomes
Type the full path of the directory where the reference genomes and indexes were located. For default, enter none

VHSSL_REF: Default
Will add more options in the future

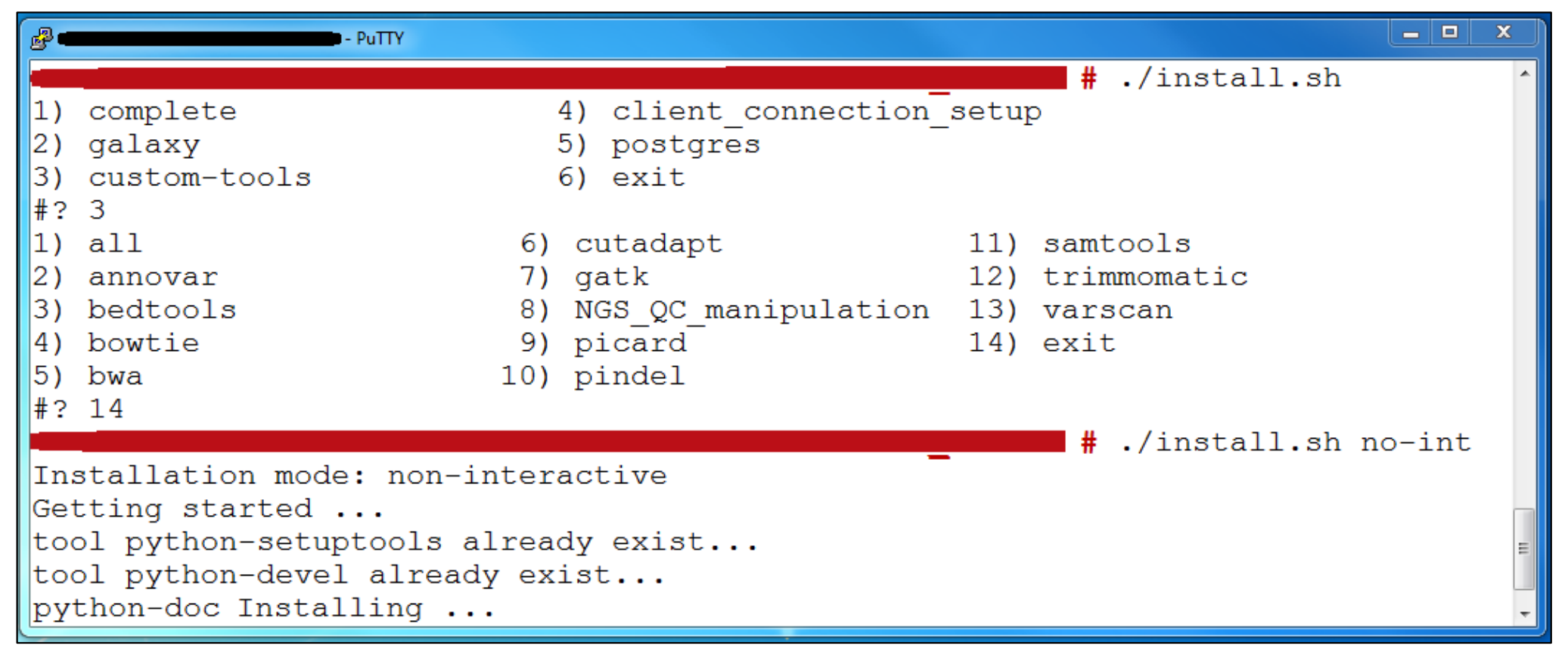
VH_REF: Default
Will add more options in the future

REPO_TOOLS: /path/to/tool_repositoires

REPO_TOOLS_WRAPPERS: /path/to/wrapper_repositories

➔ Scripts for uninstalling tools are partially available (on going)

Configuration file setup: Command line dialog



```
[redacted]# ./install.sh
1) complete                4) client_connection_setup
2) galaxy                  5) postgres
3) custom-tools            6) exit
#? 3
1) all                   6) cutadapt              11) samtools
2) annovar               7) gatk                  12) trimmomatic
3) bedtools              8) NGS_QC_manipulation   13) varscan
4) bowtie                9) picard                14) exit
5) bwa                  10) pindel
#? 14
[redacted]# ./install.sh no-int
Installation mode: non-interactive
Getting started ...
tool python-setuptools already exist...
tool python-devel already exist...
python-doc Installing ...
```

SLES issues:
- Python versions (SLES @ v2.6 vs. e.g. QIIME @ v2.7)
- Java (IBM vs. Oracle package)
- Apache modules
- …

➔ Use SLES dev repos and compile a lot (home scheme)

Tool level issues:
- Many tools for one task (and many insufficient)
- Incompatibilities esp. between toolboxes
- Disregarded standards
- Missing or bugged wrappers
- Versioning & control
- Outputs: HTML-based download links (no further data objects available for processing)
- …

➔ be careful & expect to write many snippets on your own

Pipelines:
- Porting from command line to Galaxy may get fiddly
- Custom script may make some unexpected additional work (if the respective tasks cannot be covered by stock tools)

➔ take your time
➔ for prototyping and early work stick to approved pipelines (in-house, Broad Institute etc.)

References:
- versioning also here difficult
- custom indeces: download vs. self-built

➔ read the docs carefully