A validation system for data preservation in HEP

- motivation
- concepts and design
- walk through the implementation
- summary and outlook

Yves Kemp (DESY IT), Marco Strutz
& Hermann Heßling (HTW Berlin)
ACAT 2011
Brunel University, 6.9.2011

HELMHOLTZ ASSOCIATION

DPHEP Study Group for Data Preservation and Long Term Analysis in High Energy Physics

DESY

Study Group for Data Preservation and Long Term Analysis in High Energy Physics

ICFA Study Group on Data Preservation and Long Term Analysis in HEP

- High Energy Physics experiments initiate with this Study Group a common reflection on data persistency and long term analysis in order to get a common vision on these issues and create a multi-experiment dynamics for further reference.
- The objectives of the Study Group are:
  - Review and document the physics objectives of the data persistency in HEP.
  - Exchange information concerning the analysis model: abstraction, software, documentation etc. and identify coherence points.
  - Address the hardware and software persistency status.
  - Review possible funding programs and other related international initiatives.
  - Converge to a common set of specifications in a document that will constitute the basis for future collaborations.
- Since August 2009, the Study Group is endorsed by ICFA (International Committee for Future Accelerators).

More information: Poster #59 by Roman Kogler: Data Preservation in High Energy Physics

Taken from http://www.dphep.org/

DESY

Conservation of data … and conserving analysis capability

- You need to conserve the data … that is a field of its own
- … but data alone is worthless: You also need to conserve the ability to use it, to perform analysis on it
- How to do this? Depends on the duration. Comparison with **“pizza preservation”:**

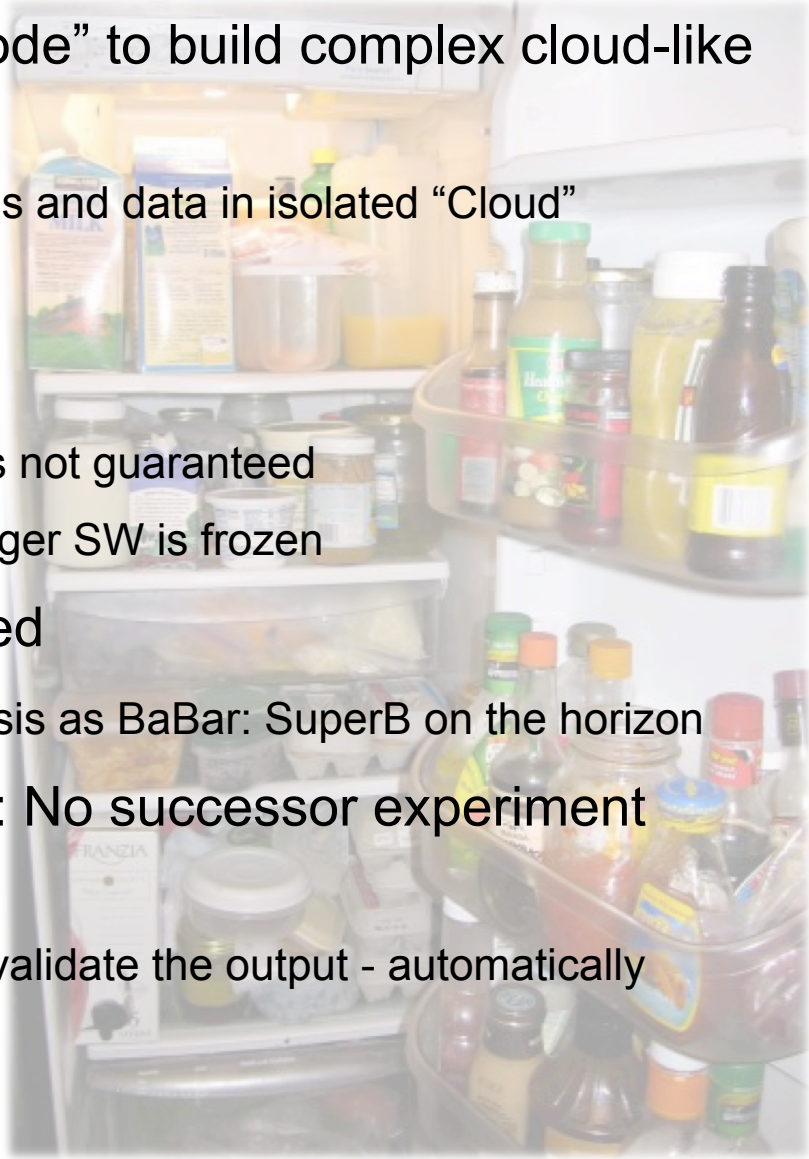
How to preserve a pizza?

- Couple of days
  - Fridge
- Couple of month
  - Deep freezer
- Couple of years???
  - Preserve the recipe
  - Practice it often: You will not forget the recipe and you can detect variations in external dependencies

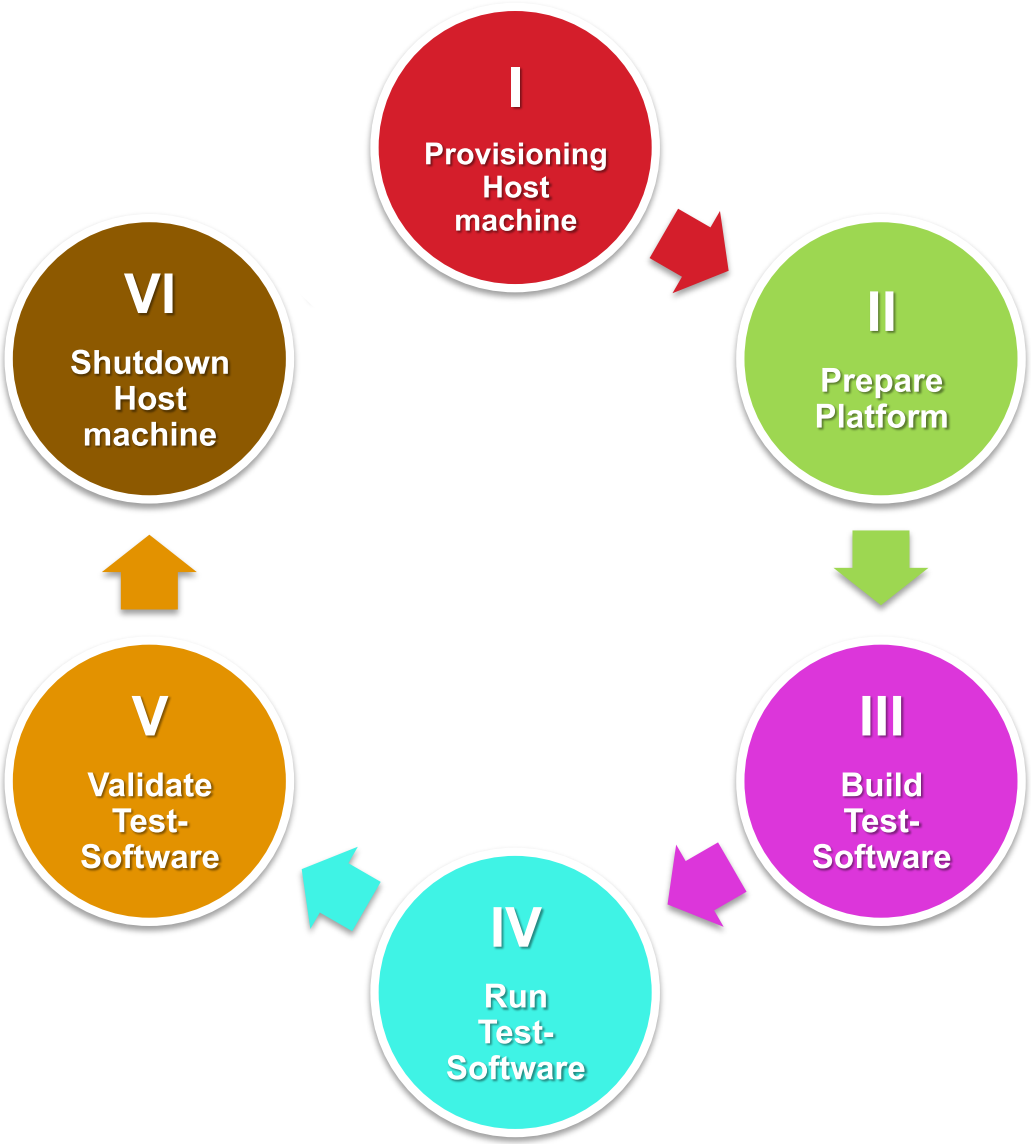
DESY

Putting software in the fridge or in the deep freezer

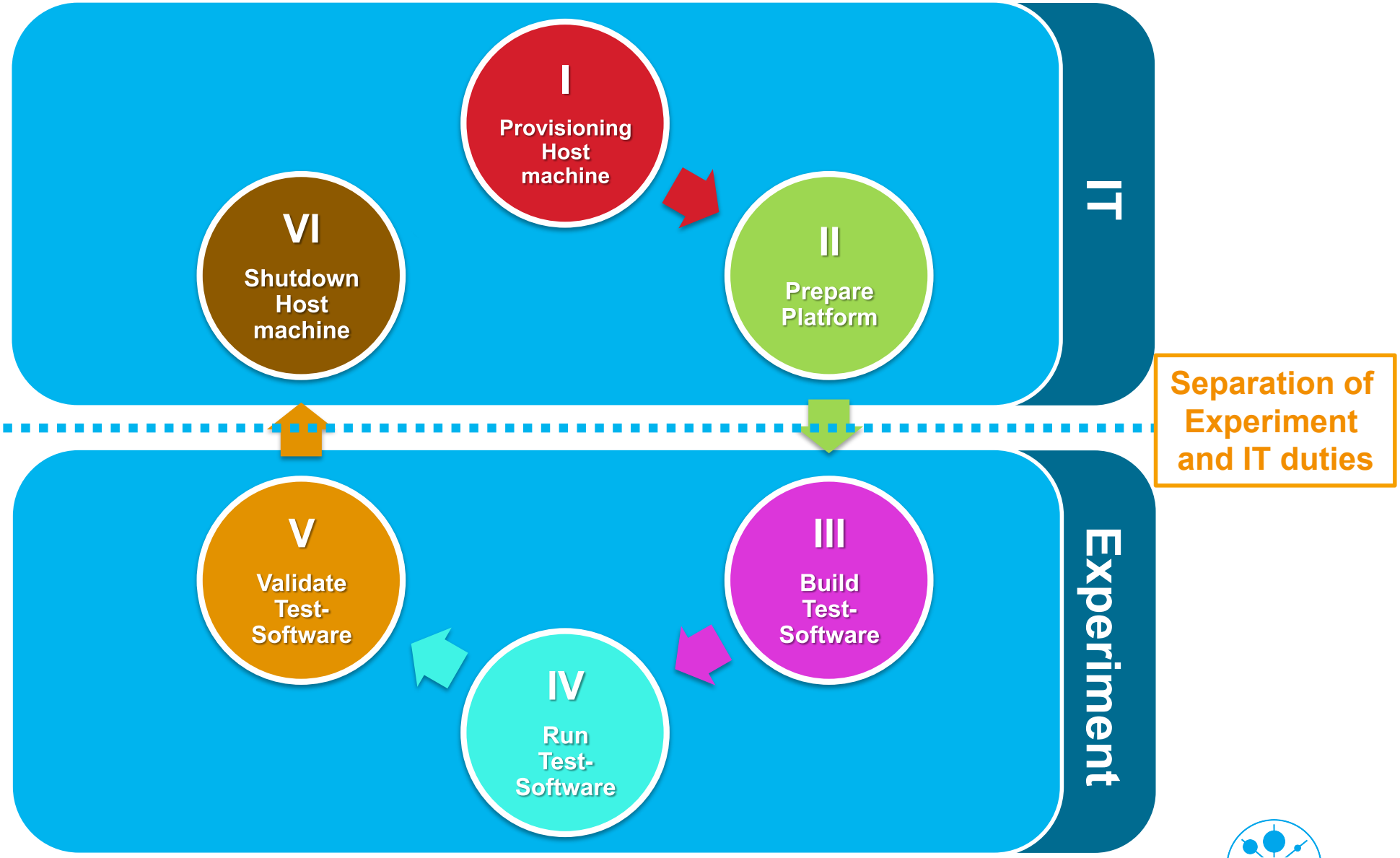
- How? Ranges from just “saving the source code” to build complex cloud-like virtualization production frameworks
  - E.g. BaBar: Having a dedicated cluster at SLAC, VMs and data in isolated “Cloud”
- Pro’s and con’s … personal summary
  - \+ Easy to do (manpower), easy to do (time)
  - – Operability of the software and correctness of results not guaranteed
  - – Changes if needed will become more difficult the longer SW is frozen
- Freezing SW OK if timeline and scope reduced
  - E.g. makes perfectly sense for BaBar SW and analysis as BaBar: SuperB on the horizon
- … but this is probably not the case for HERA: No successor experiment foreseen
  - So, cook the same recipe ever and ever again, and validate the output - automatically



The Generic Recipe (Atomic Test Life-Cycle)



… and the two cooks



… and then the automation

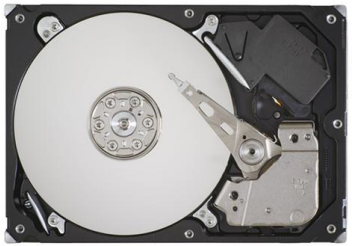
- For each configuration: Run this test cycle often
  - Sverre Jarp (5.9.2011): Don't give me a better result - just give me the same result
- You will soon detect when things break e.g.
  - A needed library is no longer available in the distro
  - SW does not compile anymore because of some update
  - SW does not run: Internal error e.g. some API changed
  - SW does not run: External error e.g. Access to mass storage changed
  - SW validation fails: Internal error e.g. compiler optimization behaves different
  - SW validation fails: External error e.g. new chip generation computes different
- You can run daily tests by hand … but easier to use virtualization

The coffee-mill idea

H1 Software
Zeus Software
$EXP Software

ROOT, GEANT,…
External SW

SL5/SL6/Debian/…
IT provides VMs



Test OK

OS lib missing
➔ IT

Tracking code error
➔ EXP-SW

Data unreadable
➔ IT & EXP-SW

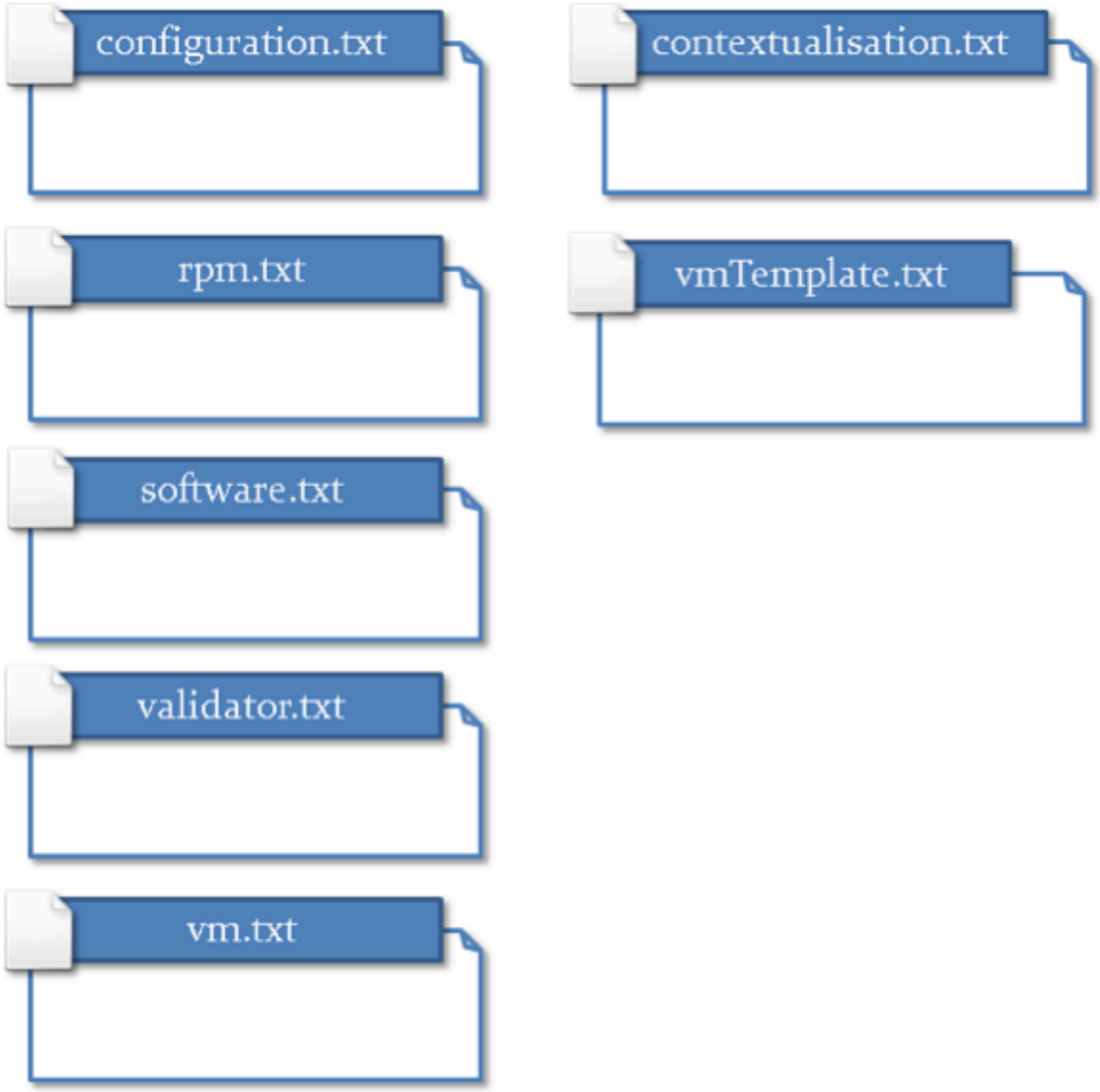
… not just an idea: Implementation by Marco Strutz (HTW Berlin) during his master thesis
Being used by H1, HERMES and ZEUS … pre-production/alpha phase

… and a walk through the system developed at DESY

- The code:
  - E.g. Some ROOT internal test
- A build.sh script
  - E.g. compile ROOT
- A run.sh script
  - E.g. Run ROOT internal Stress Test
- A validation.sh script
  - E.g. Validate the output of the Stress Test
- Additional packages in the VM image
  - E.g. gcc in version 4.N.N
- Information about the desired VM image
  - E.g. SL5.N 64bit

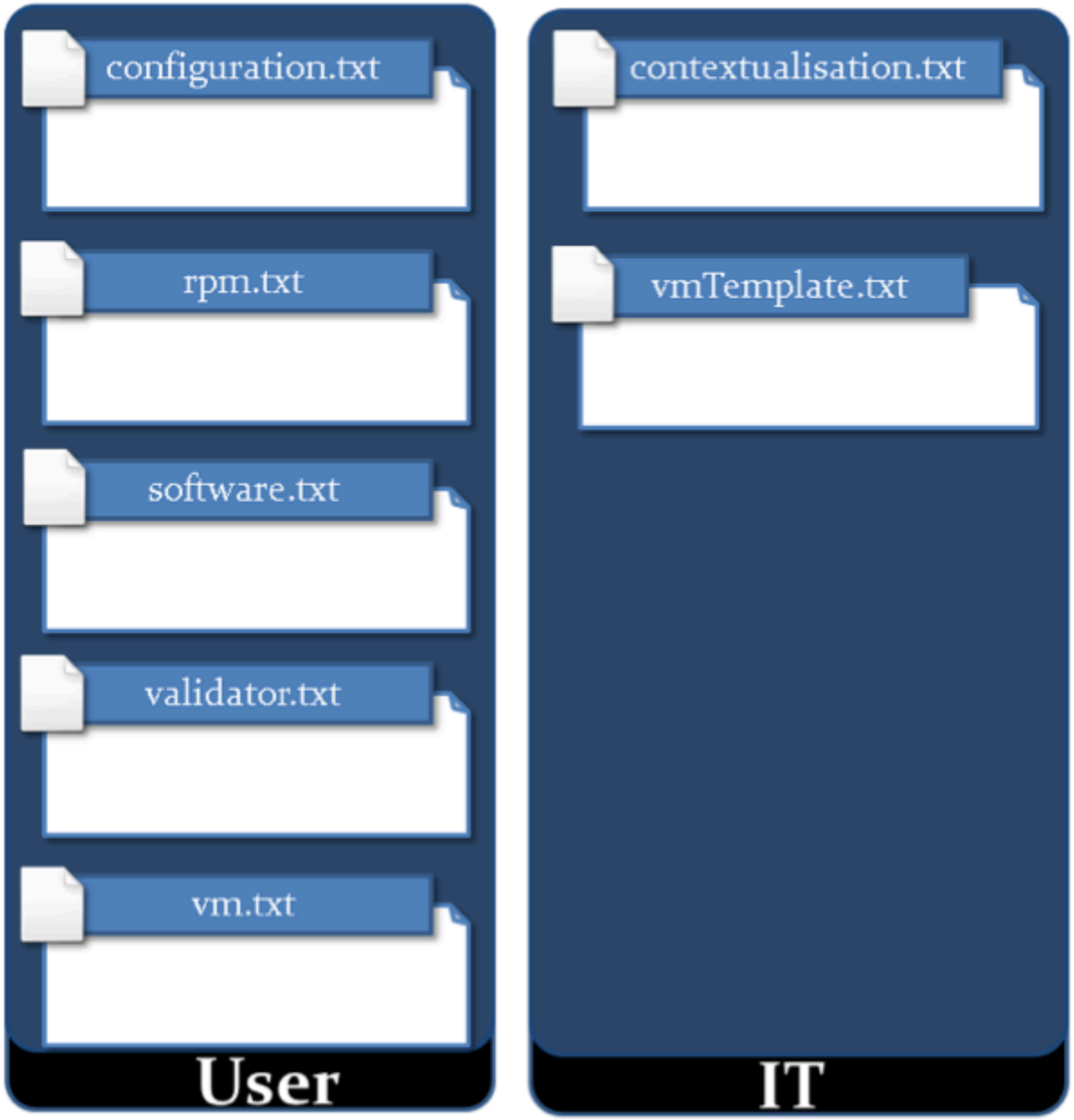
Configuration Example for ROOT

Build Configuration Files



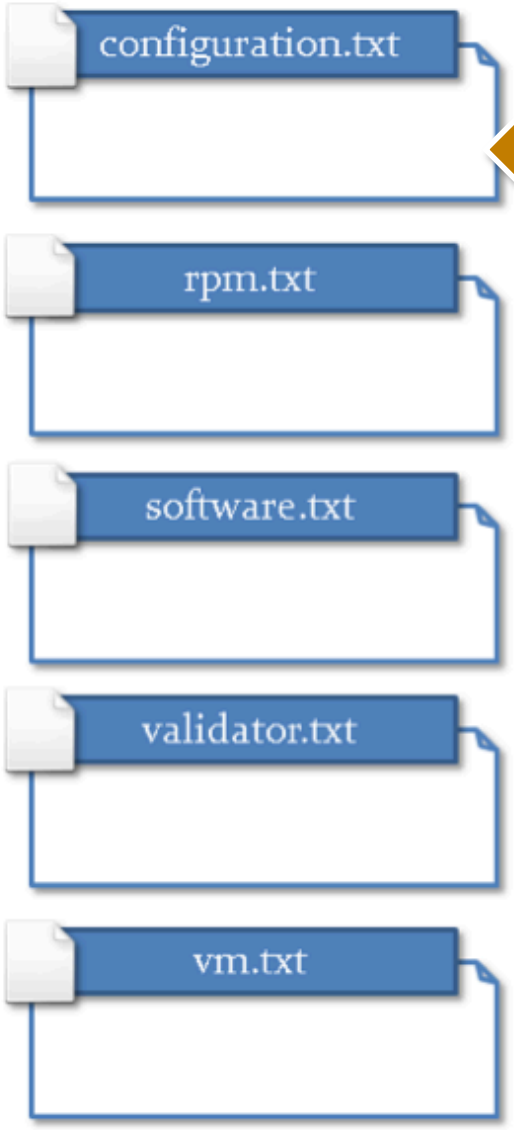
Configuration Example for ROOT

Build Configuration Files



Configuration Example for ROOT

Configuration-Files Content

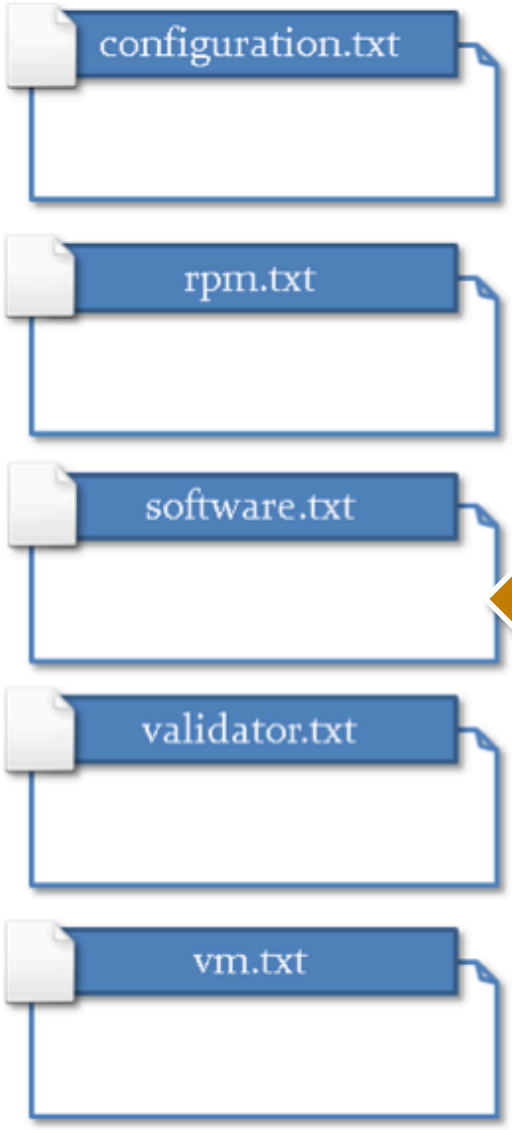



```
{"testcollection":
   {"name"        : "ROOT compiling test",
    "description": "ROOT compiling test",
    "owner"       :
       {"name"   : "marco",
        "email"  : "marco@localhost.com"}
   }
}
```

Configuration Example for ROOT
Configuration-Files Content


```
{"packages":
  {"gcc-c++" :
     {"version" : "4.1.2",
      "arch"    : "x86_64",
      "summary" : "C++ support for GCC"},
   "libX11-devel" :
     {"version" : "1.0.3",
      "arch"    : "x86_64",
      "summary" : "X.Org X11 libX11 development
                  package"},
   "libXft-devel" :
     {"version" : "2.1.10",
      "arch"    : "x86_64",
      "summary" : "X.Org X11 libXft development
                   package"},
   "libXpm-devel" :
     {"version" : "3.5.5",
      "arch"    : "x86_64",
      "summary" : "X.Org X11 libXpm development
                   package"},
   "libXext-devel" :
     {"version" : "1.0.1",
      "arch"    : "x86_64",
      "summary" : "X.Org X11 libXext development
                   package"}
   }
}
```

Configuration Example for ROOT
Configuration-Files Content

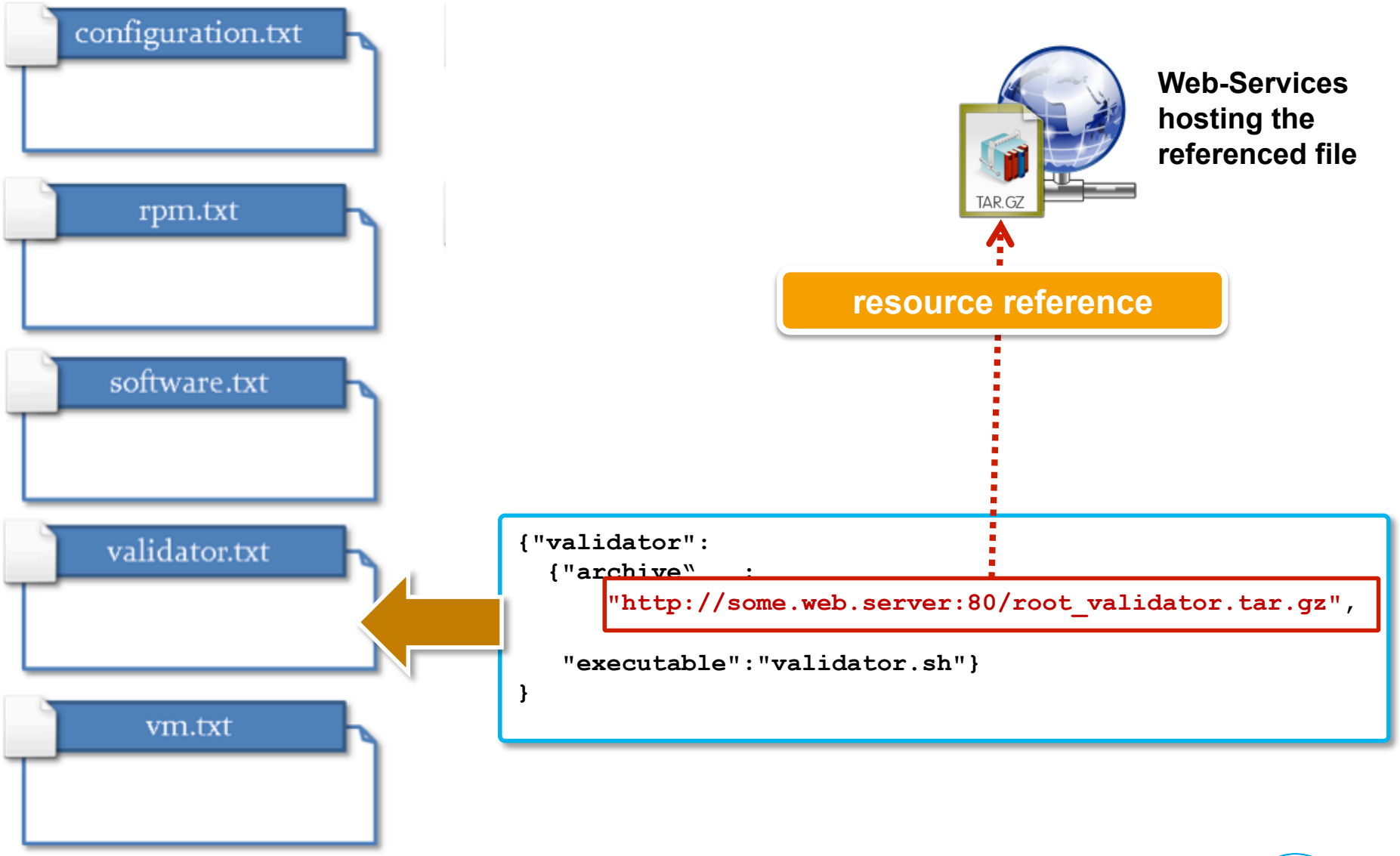


Web-Services hosting the referenced file

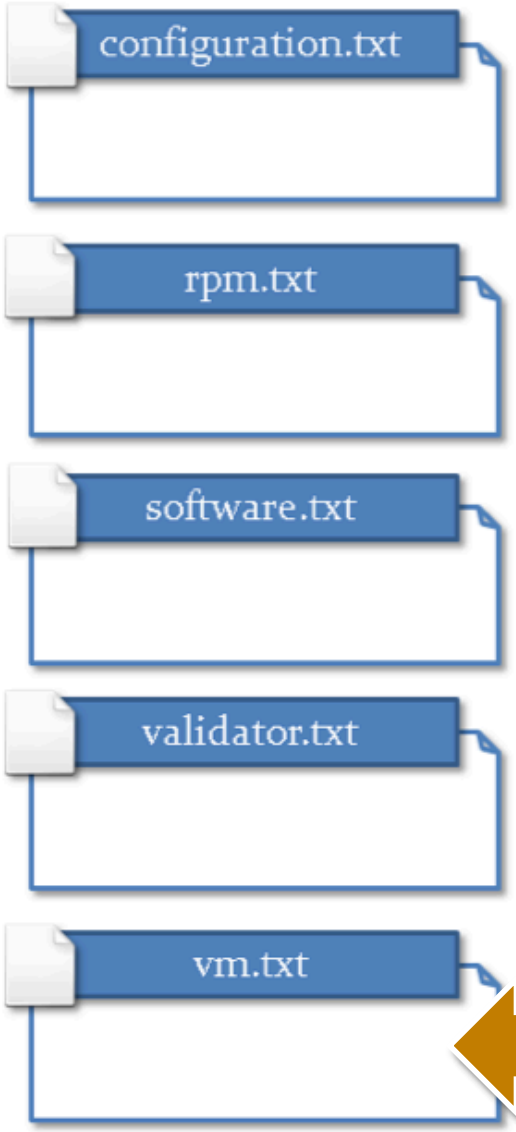
resource reference

```
{"experiment_software" :
  {"archive"    :
          "http://some.web.server:80/root_sptest.tar.gz",

   "builder"    : "build.sh",

   "executable": "run.sh"}
}
```

Configuration Example for ROOT
Configuration-Files Content



Configuration Example for ROOT
Configuration-Files Content

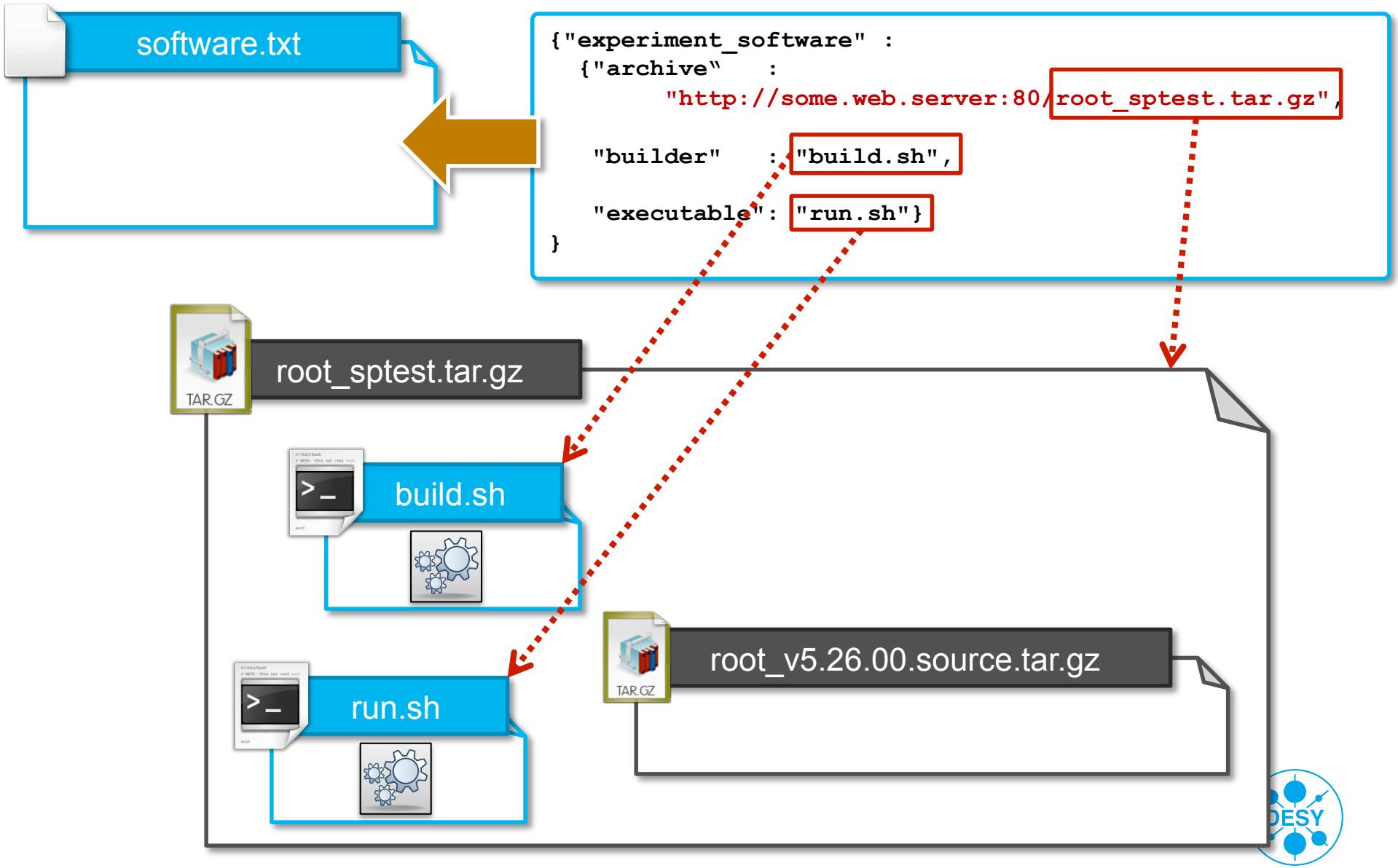


Distribution	Architecture	Description	Available Disk-Space	VM-Template ID	vm.txt
SL 6.0	i386	ScientificLinux 6.0 32bit - v0.1	9 GB	345	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/345","type":"small"}}
SL 6.0	x86_64	ScientificLinux 6.0 64bit - v0.1	9 GB	349	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/349","type":"small"}}
SL 5.5	i386	specially prepared for H1 software - v0.4	6 GB	259	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/259","type":"small"}}
SL 5.5	i386	basic installation - v0.3.1	7 GB	436	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/436","type":"small"}}
SL 5.5	x86_64	basic installation	2.5 GB	75	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/75","type":"smallWithStorage"}}
SL 5.5	x86_64	basic installation	0.2 GB	46	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/46","type":"small"}}
Debian 5	x86_64	basic installation - v0.3	0.3 GB	226	{"virtual_machine": {"template":"http://grid-lab024:18001/cloud/templates/226","type":"small"}}

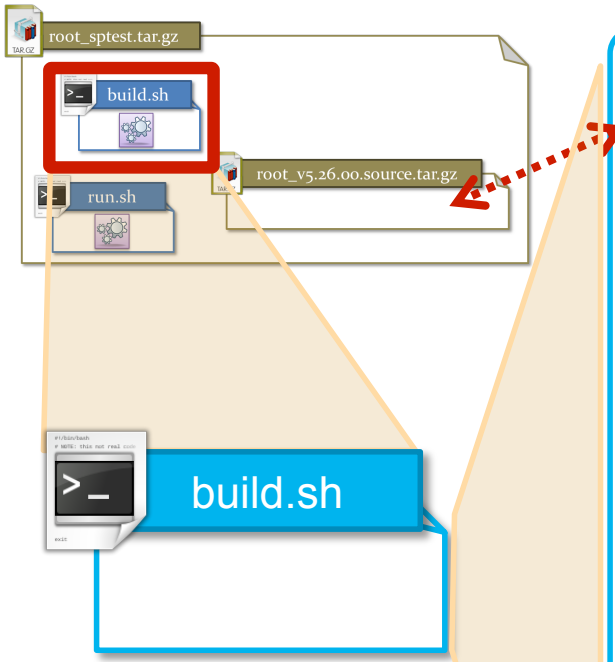
```
{"virtual_machine" :
   {"template" : "http://grid-
        lab024:18001/cloud/templates/349",
    "type":"small"}
}
```

Configuration Example for ROOT

Test-Logic Reference



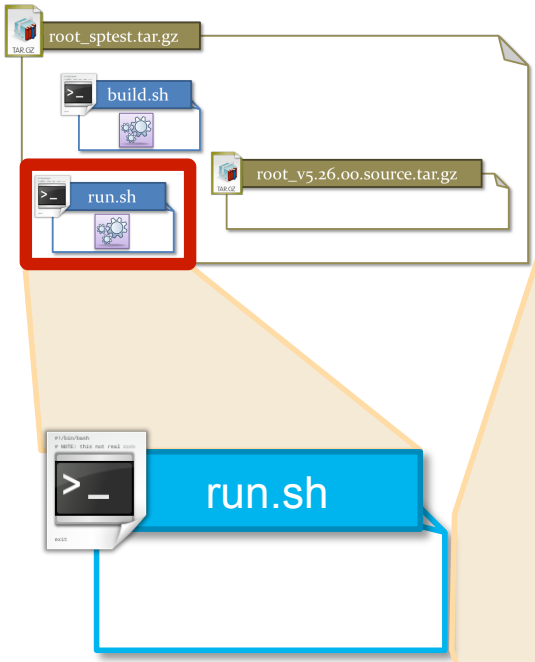
Configuration Example for ROOT
Script Payload




```
#!/bin/sh
PACKAGE="root_v5.26.00.source.tar.gz"
TARGET=./rootSrc

mkdir $TARGET
cd $TARGET

echo "extracting '$PACKAGE'..."
tar xvzf ../$PACKAGE"

#set env
export ROOTSYS=$(pwd)/root
ROOTSYS=$(pwd)/root

#configure ROOT
cd $ROOTSYS
./configure linuxx8664gcc

#make ROOT
make
```

Configuration Example for ROOT

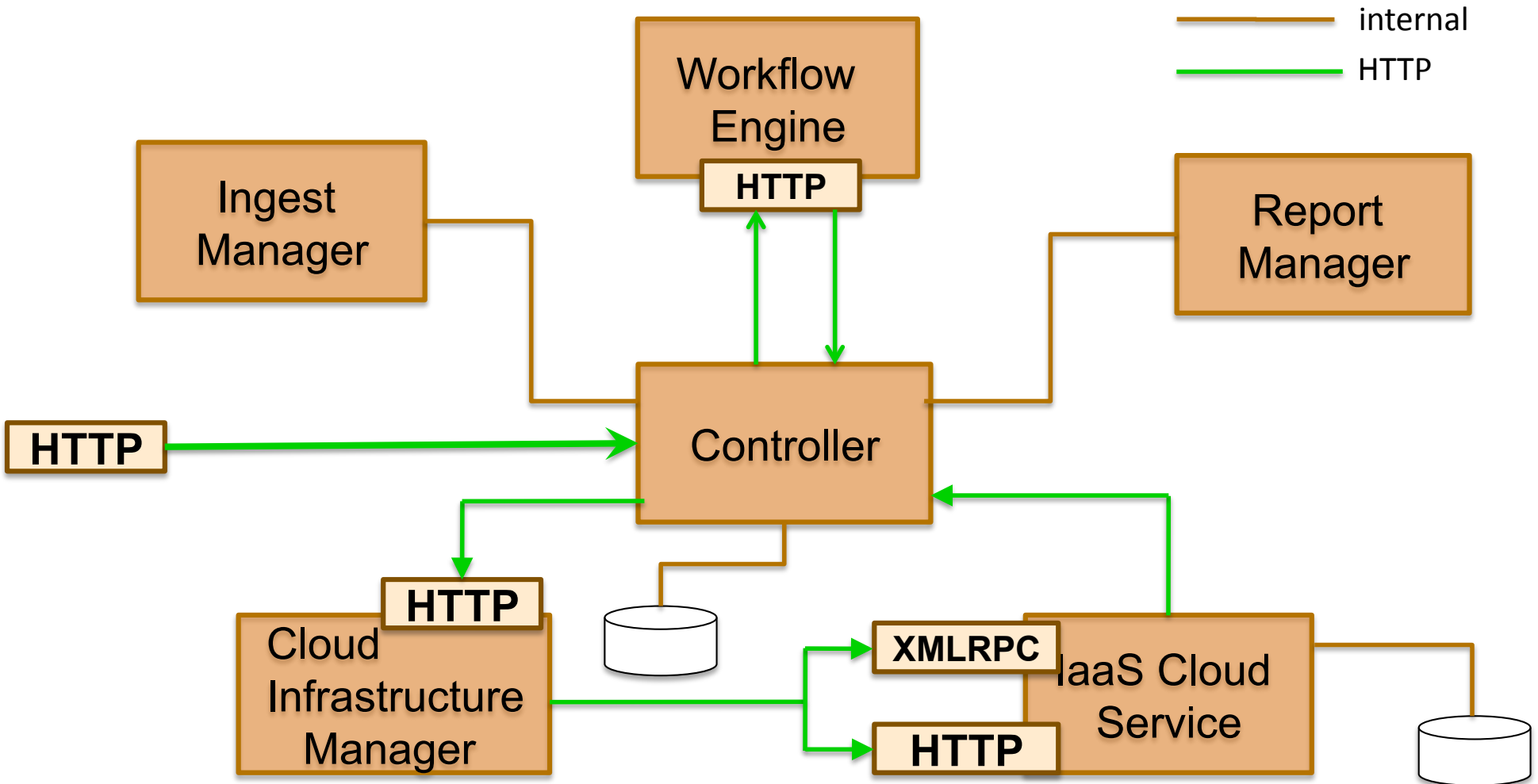
Script Payload




```
#!/bin/sh

export ROOTSYS=$(pwd)/rootSrc/root
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:
$LD_LIBRARY_PATH

cd ${ROOTSYS}/test/

#part of the run-step can also be a build-
call
make

echo "running 'stressHepix' test..."
./stressHepix

echo "running 'bench' test..."
./bench
```

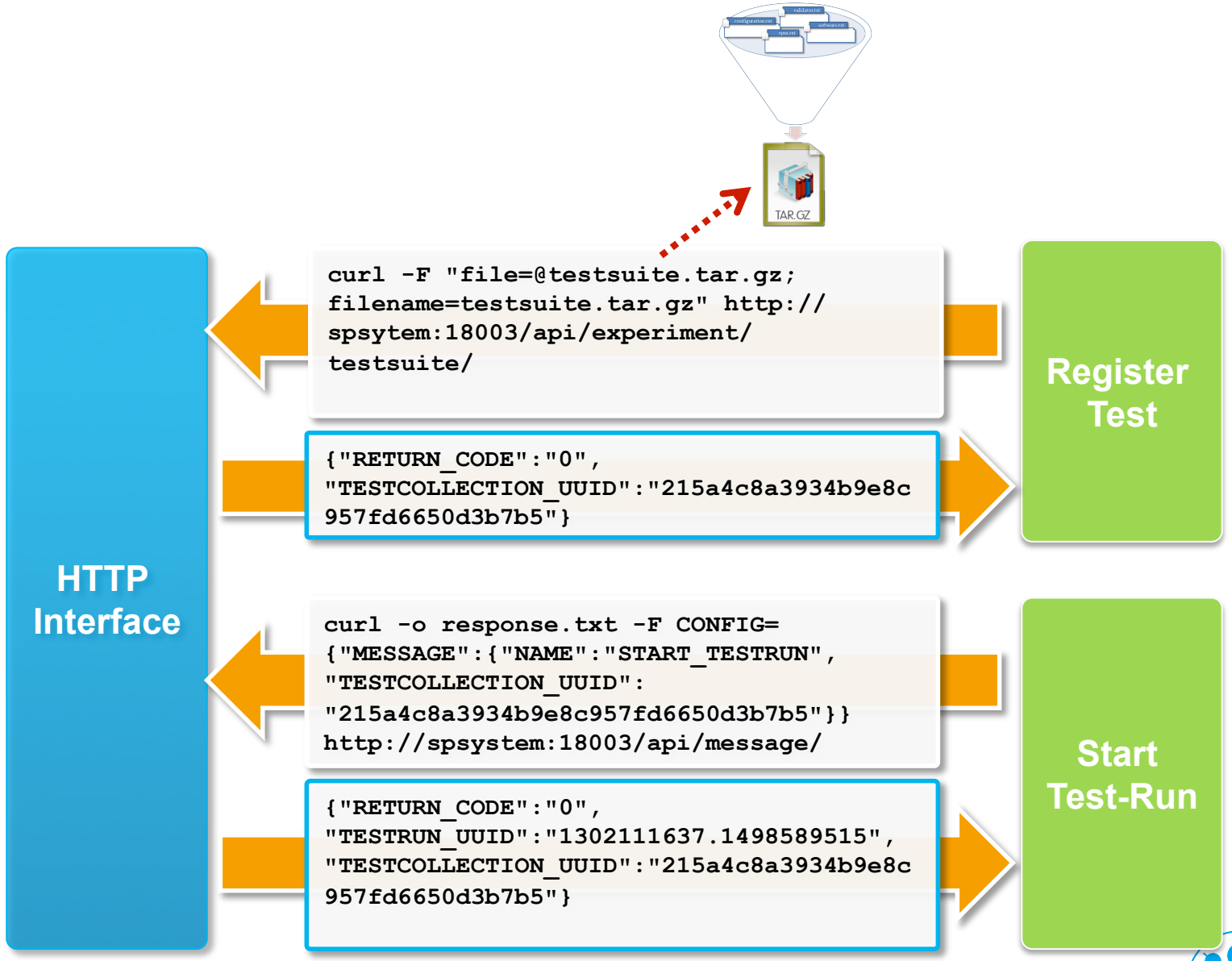
A Parenthesis: Components and Communication

- Modular design of the Validation Framework
- Communication based mostly on HTTP

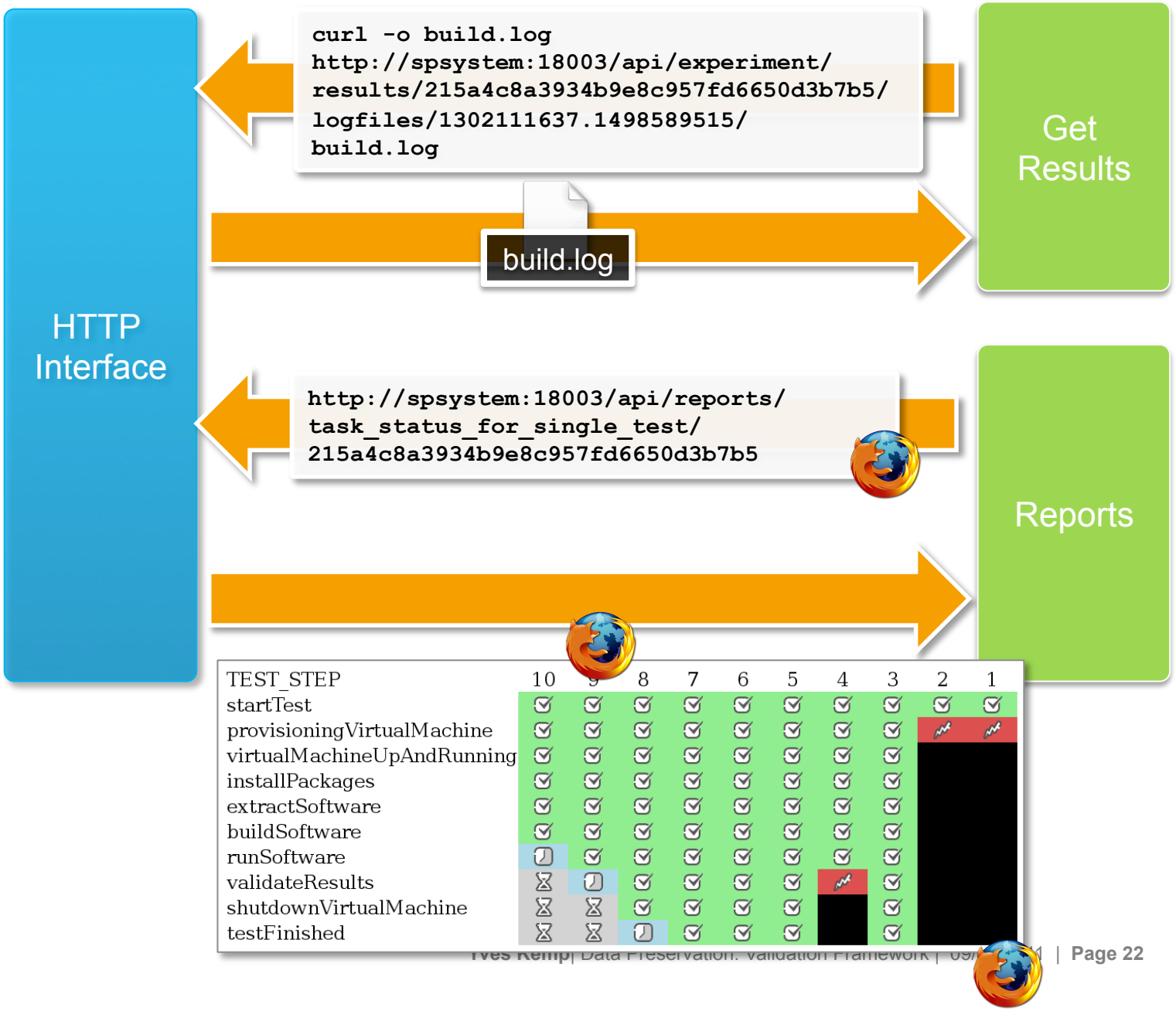


Structure “à la OAIS”

Launch a test in the Validation Framework



Get results from the test run



Get results from the test run

```
curl -o build.log
http://spsystem:18003/api/experiment/
```

```
+ wget -q http://www.desy.de/~johndoe/virt/sw-test.tgz
+ tar xvzf sw-test.tgz
tar: sw-test.tgz: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
tar: Child returned status 2
tar: Exiting with failure status due to previous errors
+ chmod 755 swmc_run
chmod: cannot access `swmc_run': No such file or directory
+ ./swmc_run config1234 seed 9876
/home/dphep/application.sh: line 9: ./swmc_run: No such file or directory
+ ls -ltra
total 32
-rw-r--r--. 1 dphep dphep  124 Mar 31  2010 .bashrc
-rw-r--r--. 1 dphep dphep  176 Mar 31  2010 .bash_profile
-rw-r--r--. 1 dphep dphep   18 Mar 31  2010 .bash_logout
drwxr-xr-x. 2 dphep dphep 4096 Mar 31  2010 .gnome2
drwxr-xr-x. 4 dphep dphep 4096 May 13  2010 .mozilla
drwxr-xr-x. 3 root  root  4096 Jun  7  2010 ..
drwx------. 4 dphep dphep 4096 Mar 19 19:04 .
-rwxr-xr-x. 1 dphep dphep  167 Mar 19 19:04 application.sh
```

“Stress tests”

- Also “stress test” your experiment software!
- What to test? We (IT guys) do not know – but we know some things that failed in the past, and which one should write tests for:
  - Is the access to the mass storage working? E.g. dCap library, door name and port, …
  - Are external services still running and working? (Databases, CVS,…)
  - If you compile SW: Does it compile at all with the current update of your compiler?
  - If you compile SW: Is the compiler optimization doing the same things that before?
  - Do OS tools behave the same way?
  - If there is a change in underlying HW architecture: Are computation results the same?
  - …
- Basically, these are just tests that already exist for e.g. validating nightly builds or validating a Grid SW installation or …

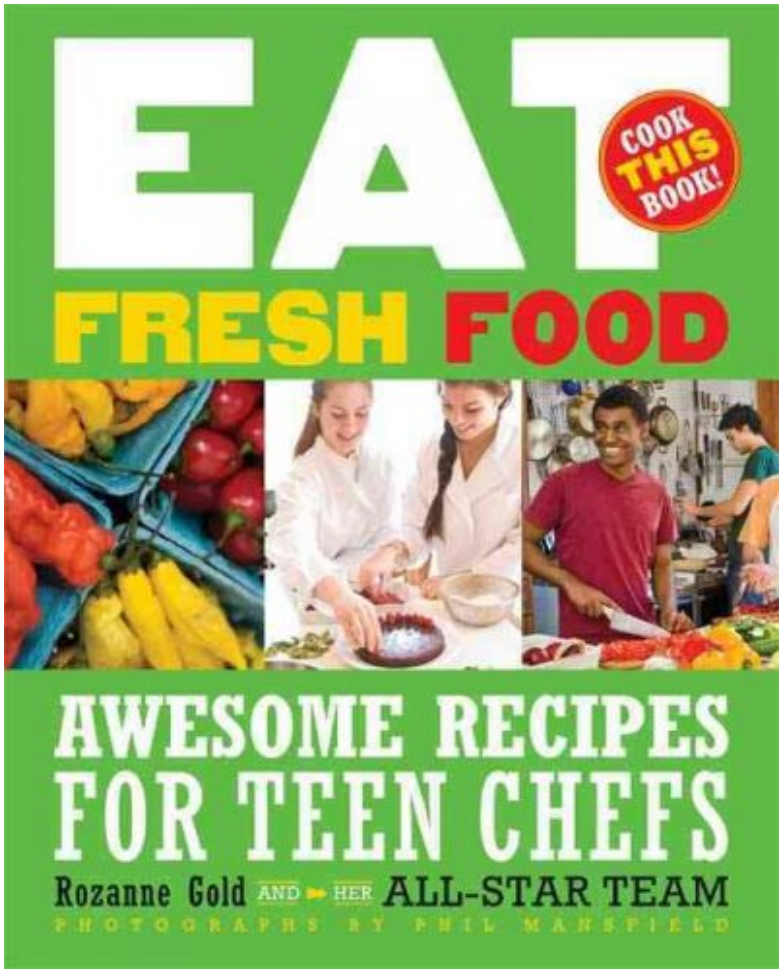
Things we cannot do in the validation framework

- The framework is designed for software verification, validation and migration support only.
- It is not designed for mass production or large scale analysis
  - Both in HW resources and in the interface

- The framework tells you whether you ***could run production at a particular moment*** – and it can tell you how to prepare your system ("the pizza recipe")
- If you ***want to run production at a particular moment***, use other resources: Anything that fits then: Institute Cluster, Grid, Cloud, Sky, Quantum Computer,…

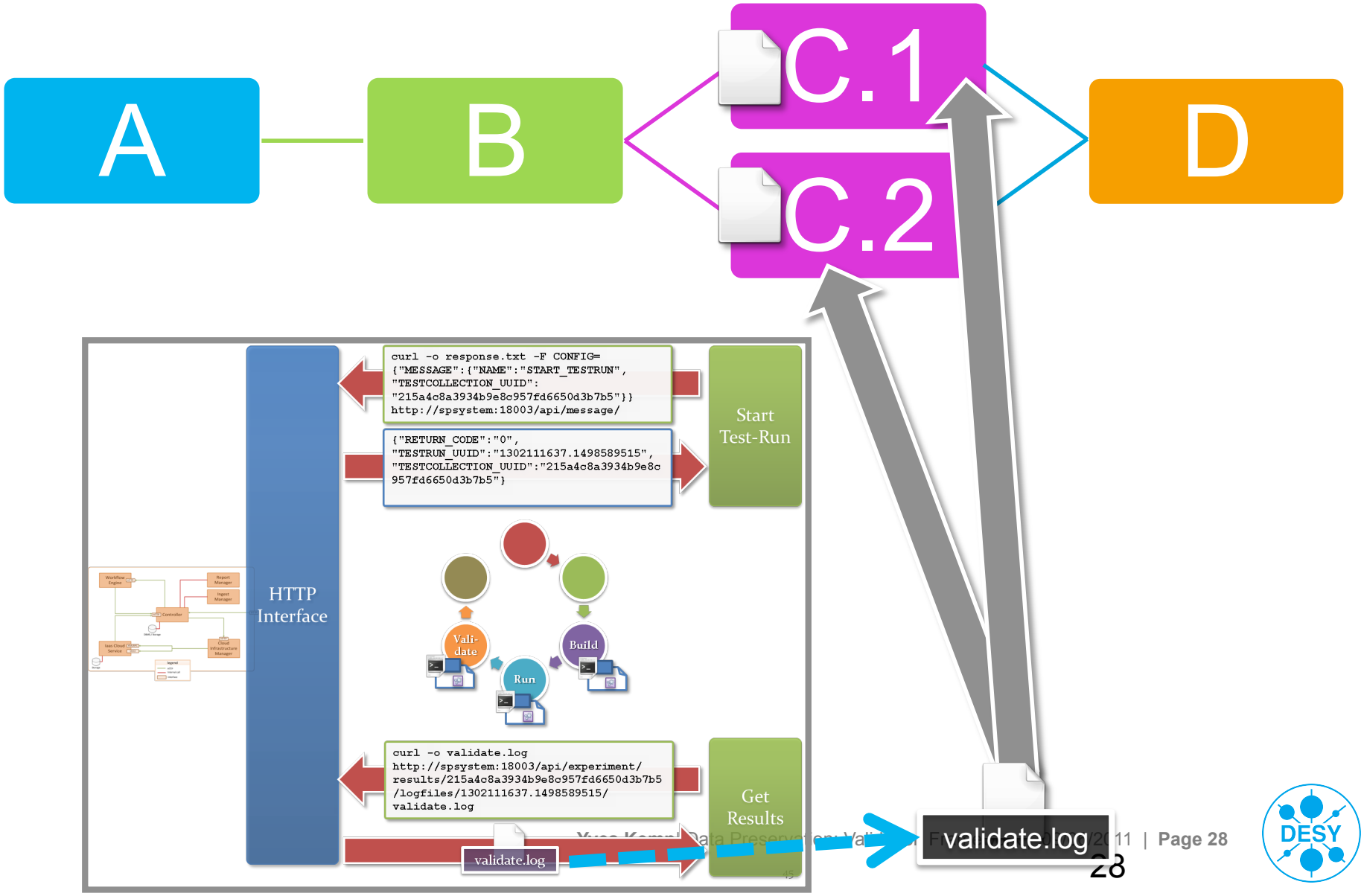
Conclusion and outlook

- DESY IT developed a system for automated validation of software in DPHEP context
  - Master thesis of Marco Strutz
- Have moved to production hardware, still some polishing needed
  - Doing together with brave local HERA users
  - Consider project still in preproduction/alpha state
  - … but moving forward!
- Validation means running tests
  - These must be provided by experiments … the more the better!



Backup material – technical details

Integration into complex or external workflows possible



Software Components

- Controller, [Cloud Infrastructure | Ingest | Report] Manager
  - Logic : Python v2.4.3 (compatible to v2.6.5)
    http://www.python.org/download/releases/2.4.3/
  - Database : SQLite v2.8.17
    http://www.sqlite.org/
  - Web-Services : web.py 0.34 (Python Framework)
    http://webpy.org/
- Workflow Engine
  - Hudson CI v1.386 (2010/11/19) -> Update to Jenkins planned
    http://hudson-ci.org/changelog.html
- IaaS Cloud Service
  - OpenNebula v2.2 (Aug 8 2011) (mostly written in Ruby)
    http://opennebula.org/
  - Hypervisor: KVM (qemu-kvm-0.12.3)

Hardware and Controller setup

- Controller:
  - For hardware consolidation purpose run in a XEN enterprise cluster
  - Two distinct machines / SL5

- Cloud backend: OpenNebula [1xFront-End | 2xCluster Nodes]
  - Current DELL based machines
  - Front-end: Has 1.5 TB fast disk array for managing VM images
  - 1 Cluster node with Intel CPU, 1 Cluster node with AMD CPU
  - … can easily be expanded – or integrated into “Your Own Cloud”

Communication Protocols

- Validation Framework Interface
  - JSON
    http://www.json.org/
  - RESTful WebService
  - Linux Shell (/bin/sh)
- Cloud Infrastructure Manager → OpenNebula
  - OCCi (Open Cloud Computing Interface)
    http://occi-wg.org/
  - XMLRPC
    http://www.xmlrpc.com/
- OpenNebula
  - Control Plane: SSH + libvirt (http://libvirt.org/)
  - VM Image Access: NFS