NORDUGRID

NORDUGRID-D2.5-2 ???
28/11/2009

DYNAMIC RUNTIME ENVIRONMENT INSTALLATION WITH JANITOR

The Janitor and this document with it is under continuous development. Your comments and suggestions are appreciated.

Michael Glodek, Daniel Bayer, Steffen Möller[*]

---

[*] moeller@inb.uni-luebeck.de

Contents

1 Introduction

The *Janitor* is a service for the automated installation of runtime environments for grid computing elements. Its command line interface allows for a direct interaction with site administrators. However, the main stimulus for its development was the idea integrate such a service with the regular handling of compute jobs. For ARC this is performed by the A-REX module.

From the programmer's view, the Janitor is mostly a Perl script and the routines within A-REX to invoke it. The site administrator will also associate with it also the Catalog files, that describe the availability of runtime environments, and the repository of installable runtime envionments themselves, which are regular tar archives obeying to particular structure and reside in a separate folder. In order to minimise the latency for the invocation of the Perl script and the associated parsing of files, a Janitor web service was developed, which still is a Perl script.

1.1 Motivation

A major motivation for grid projects is to stimulate new communities to adopt computational grids for their causes. From the current grid user's viewpoint, the admission of users of a very different scientific disciplines or compute skills will impose difficulties in the communication between site maintainers. One will not even understand the respective other side's research aims. Hence, the proper installation of non-standard software (read Runtime Environments) is not guaranteed.

A core problem remains to distribute a locally working solution, the Know-How, quickly across all contributing sites, i. e., without manual interference. Every scientific discipline has its respective own set of technologies for the distribution of work load. For instance, research in bioinformatics requires access to so many different tools and databases, that few sites, if any, install them all. Instead, the use of web services became a commodity, with all the problems with respect to bottlenecks and restrictions of repeated access. The EU project KnowARC[*], amongst other challenges, with the here presented work extends the NorduGrid?s Advanced Research Connector (ARC) grid middleware [?] towards an the automated installation of software packages.

An automation of the software installation, referred to as dynamic Runtime Environments, seems the only approach to use the computational grid to its full potential. Components of workflows shall be spawned as jobs in a computational grid using dynamic Runtime Environments rather than as shared web services. The grid introduces an extra level of parallelism that web services cannot provide. The required short response times and the heterogeneous education of site-administrators on a grid demand an automatism for the installation of software and databases without manual interference [?].

1.2 Overview

This document will start with a chapter of how to set up janitor locally. The following chapter will give an instruction on how to use Janitor with A-REX and/or without A-REX . Afterwards, in the third chapter, the maintenance of the program will be presented, which is basically covering the preparation of runtime environments. Deeper knowledge about the design of the Janitor will be given by the subsequent forth chapter. In the fifth chapter, an outlook on ongoing or future developments will presented.

---

[*] http://www.knowarc.eu

2 Installation

The Janitor depends on two perl packages listed in table 2.1. To have the WebService interface for Janitor, the packages listed in the table 2.2 as build requirements.

Table 2.1: Required perl packages for the Janitor. Log4perl is used for the internal logging of Janitor, while the Redland RDF libraray is used for accessing the knowledge base of RTEs.

liblog-log4perl-perl	*Log4perl is a port of the log4j logging package*
librdf-perl	*Perl language bindings for the Redland RDF library*

Table 2.2: Optional libraries for Janitor. The library libperl-dev provides header files which are needed to link the WebService to the Perl interpreter.

libperl-dev	*Perl library: development files*

If you are using regular Debian or Ubuntu packages, then the Janitor can be installed as root by "apt-get install nordugrid-arc1-janitor". Installing it will not drag other components of ARC with it since the Janitor can be used in its own right or in conjunction with another grid system, possibly. Packages for Redhat/Fedora and SuSE/OpenSuSE are also provided, the redland library however may not yet be available for those systems. If you are compiling the sources yourself, the Janitor will be enabled by default. If desired, it is possible to disable it with the *configure* flags *–disable-janitor-service* for the complete janitor or *–disable-janitor-webservice* for only the Web Service support.

Furthermore it is recommended to install the ontology editor Protègè[*] in order to be to easily maintain the knowledge database of installable packages.

2.1 Configuration

The current version of Janitor can be configured using the common file *arc.conf* which is to be found in the configuration directory *etc*. Janitor is using the environment variable *NORDUGRID_CONFIG* to determine the location of the corresponding file. If the variable is not set, the default location */etc/arc.conf* will be used. The configuration is assigned by the section [janitor]. The table 2.3 contains the available tags for janitor configuration.

---

[*] http://protege.stanford.edu

Table 2.3: Tags usable in *arc.conf* within the section janitor. Tags usable in *arc.conf* within the section janitor.

tag	example	description
enabled	”1”	Boolean flag which enables or disables janitor in A-REX.
uid	”root”	The effective uid.
gid	”0”	The effective gid.
registrationdir	”/var/spool/nordugrid/janitor”	Directory where we the current states of jobs are kept.
catalog	”/var/spool/nordugrid/janitor/catalog/knowarc.rdf”	URL of the catalog containg the package information.
downloaddir	”/var/spool/nordugrid/janitor/download”	Directory for downloads
installationdir	”/var/spool/nordugrid/janitor/runtime”	Directory for installation of packages
jobexpirytime	”7200”	If a job is older than this, it is considered dead and assigned to be removal pending.
rteexpirytime	”36”	If a runtime environment was not used for this time, it will be assigned to be removal pending.
allow_base	”*”	Allow rule for base packages.
deny_base	”debian::etch”	Deny rule for base packages.
allow_rte	”*”	Allow rule for base packages.
deny_rte	”APPS/MATH/ELMER-5.0.2”	Deny rule for base packages.
logconf	”/opt/nordugrid/etc/log.conf”	Location of the logging configuration file for janitor.

The parameter `enabled` defines whether the Janitor shall be used within A-REX or not. Use the value `"0"` to disable the Janitor. The `uid` and the `gid` are defining which effective user and group id shall be used for Janitor. Setting these to values different from root will increase security.

The `registrationdir` describes the directory in which the subdirectories `jobs` and `rtes` will be created. In these directories the states of the jobs and the runtime environments are stored. Please recall that the Janitor does not use a database as a backend, but all communication between invocations are performed via files in those folders.

The knowledge base of installable packages is specified by the parmeter `catalog`. Its value can be any kind of URL pointing to a file written in the Resource Description Framework (RDF) format. One should not light-heartedly use a remote address for this purpose. Such a remote source needs to be trusted.

The specification of the RDF file will be explained in detail in section 4.1. The parameter `downloaddir` assignes the directory in which the installation files will be saved after they have been downloaded or copied from the repository which was specified by the catalog. Please remember: the URL in arc.conf indicates the location of the catalog. And the URLs somehow specified in the catalog specify the location from where to download the runtime environment.

The `installationdir` finally specifies the directory into which all packages will be installed. This `installationdir` directory needs to be available for all computing nodes for the execution of arbitrary programs, mostl likely by using it as a shared NFS volume.

If the configuration file furthermore contains the `runtimedir` tag within the section `grid-manager`, the Janitor will also create a symbolic link in the `runtimedir` pointing to the configuration script of the installation done by Janitor. The tags `jobexpirytime` and `rteexpirytime` are used for automated cleanup and are defined in seconds. The default value for thew`jobexpirytime` is seven days and for the `rteexpirytime` three days. The additional tags `allow_base`, `deny_base`, `allow_rte` and `deny_rte` are used to include or exclude certain base packages or runtime environments of the catalog. This feature is useful, if the catalog is maintain by a higher organization. But again: you need to trust it.

The path to the log4perl configuration file is defined by the tag `logconf`. Examples how to configure arc and log4perl are provided in the Listings 2.1 and 2.2.

Listing 2.1: Example *arc.conf* settings for janitor.

```
[janitor]
enabled="1"
logconf="/opt/nordugrid/etc/log.conf"
registrationdir="/var/spool/nordugrid/janitor"
installationdir="/var/spool/nordugrid/janitor/runtime"
downloaddir="/var/spool/nordugrid/janitor/download"
jobexpirytime="7200"
rteexpirytime="36"
uid="root"
gid="0"
allow_base="*"
allow_rte="*"

[janitor/nordugrid]
catalog="/var/spool/nordugrid/janitor/catalog/knowarc.rdf"
```

Listing 2.2: Example *log.conf* settings for janitor.

```
# Master Loglevel
# [OFF | DEBUG | INFO | WARN | ERROR | FATAL]
#log4perl.threshold = OFF

log4perl.rootLogger = WARN, DebugLog, MainLog, ErrorLog
log4perl.appender.DebugLog = Log::Log4perl::Appender::Screen
log4perl.appender.DebugLog.layout = PatternLayout
log4perl.appender.DebugLog.layout.ConversionPattern = [%C] %d %p> %m%n

log4perl.appender.MainLog = Log::Log4perl::Appender::File
log4perl.appender.MainLog.Threshold = DEBUG
log4perl.appender.MainLog.filename = /var/log/janitor.log
log4perl.appender.MainLog.layout = PatternLayout
log4perl.appender.MainLog.layout.ConversionPattern = %d %p> %m%n

log4perl.appender.ErrorLog = Log::Log4perl::Appender::File
log4perl.appender.ErrorLog.Threshold = ERROR
log4perl.appender.ErrorLog.filename = /var/log/janitor_error.log

```

```
log4perl.appender.ErrorLog.layout = PatternLayout
log4perl.appender.ErrorLog.layout.ConversionPattern = %d %p> %m%n
```

2.2 Limitations

The Janitor is only available for UNIX-based servers. While not tested, it should also be functional on MacOS X and Windows with Cygwin.

3 Usage

The Janitor can be used either with or without an integration with A-REX . The Janitor's installation will not be affected by this decision pro or cons and integration with A-REX . Both can be installed in parallel. If A-REX is not allowed to install runtime environments upon demand, such automated installations can still be invoked manually via the Janitor's command line interface.

3.1 Janitor with A-REX

Runtime Environments can be specified using the supported job description languages. Representative two common languages shall be explained at this point: xRSL and JSDL. Listing 3.1 shows the xRSL example in which two runtime environments are requested.

Listing 3.1: Job submission using the xRSL job description language.

```
&
(executable = "run.sh" )
(arguments = "weka.classifiers.trees.J48" "-t" "weather.arff")
("inputfiles" = ("weather.arff" "" ))
("stderr" = "stderr" )
("stdout" = "stdout" )
("gmlog" = "gmlog" )
("runtimeenvironment" = "APPS/BIO/WEKA-3.4.10")
("runtimeenvironment" = "APPS/BIO/WISE-2.4.1-5")
```

A comprehensive reference manual of the Extended Resource Specification Language (XRSL) can be found at www.nordugrid.org/documents/xrsl.pdf [?]. Within Listing 3.2 an example using JSDL is provided. The specification of how to assign runtime environments in JSDL is currently only defined within the nordugrid jsdl-arc schema http://svn.nordugrid.org/repos/nordugrid/arc1/trunk/src/services/a-rex/grid-manager/jobdesc/jsdl/jsdl_arc.xsd.

Listing 3.2: Job submission using JSDL.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition
  xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl"
  xmlns:posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
  xmlns:arc="http://www.nordugrid.org/ws/schemas/jsdl-arc">
  <JobDescription>
    <Application>
      <posix:POSIXApplication>
        <posix:Executable>/bin/sleep</posix:Executable>
        <posix:Argument>120</posix:Argument>
      </posix:POSIXApplication>
    </Application>
    <DataStaging>
      <FileName>test.sh</FileName>
      <Source/>
      <Target/>
    </DataStaging>
    <DataStaging>
      <FileName>transferGSI-small</FileName>
      <Source>
        <URI>gsiftp://pgs02.grid.upjs.sk:2811/unixacl/transferGSI-small</URI>
      </Source>
      <Target/>
    </DataStaging>
    <Resources>
      <arc:RunTimeEnvironment>
        <arc:Name>APPS/BIO/WISE-2.4.1-5</arc:Name>
        <arc:Version><Exact>2.4.1</Exact></arc:Version>
      </arc:RunTimeEnvironment>
      <arc:RunTimeEnvironment>
        <arc:Name>APPS/BIO/APPS/BIO/WEKA-3.4.10</arc:Name>
        <arc:Version><Exact>3.4</Exact></arc:Version>
        </arc:RunTimeEnvironment>
      </Resources>
    </JobDescription>
  </JobDefinition>
```

3.2 Janitor without A-REX

On Linux systems, the Janitor's standalone commandline tool is available as /usr/lib/arc/janitor. Some Linux distributions may prefer /usr/libexec or similar paths. The script is only functional as root[*]. To find that binary directly, you may decide to add that location to your $PATH environment variable.

The available commands to the Janitor, implemented as options to the janitor script, are listed in the Table 3.1. The most important commands for Janitor are `register`, `deploy` and `remove`. In order to register a job along with a set of runtime environments in the Janitor, the first command `register` followed by a job identifier and a list of runtime environments has to be used. A job is identified by sequence of numbers. Runtime environments are specified by a string containing the name as it is defined within the catalog (resp. the runtime directory of the grid-manager).

The command `deploy` extracts the necessary dependcies of the desired runtime environments and then downloads and installs the required packages.

The disassociation of a job with a runtime environment is indicated as a "removal" of that job from the runtime environment. This is performed with the Janitor the command `remove`. The command only removes the job entry and the lock on the runtime environment. If there are no more locks on the runtime environment it might be deleted for real.

Easy commandline examples are provided in Listing 3.3. You may also want to inspect the janitor(8) man page.

Every command has a certain behaviour for its exit status. The Table 3.2 lists the possible outcomes. A value of 0 always indicates that no error occurred.

Listing 3.3: Example *log.conf* settings for janitor.

```
# janitor register 1999 APP/BIO/JASPAR-CORE-1.0 APPS/BIO/APPS/BIO/WEKA-3.4.10
# janitor deploy 1999
# janitor remove 1999

# janitor sweep --force
# janitor setstate REMOVAL_PENDING APP/BIO/JASPAR-CORE-1.0 APPS/BIO/APPS/BIO/WEKA-3.4.10

# janitor search JASPAR WEKA
# janitor list
# janitor info 1999
```

Once a dynamic runtime environment is installed, it looks completely indistinguishable from traditionally installed runtime environments. This also means that the general concept to have one installation performed for all compute nodes in the network is kept.

3.3 Janitor with A-REX

The motivation to have a runtime environment available comes from the submitters of the grid jobs that depends on that runtime environment for their execution. The site administrator's sole responsibility is to have the dynamic runtime environment at the site's disposal. No more. With A-REX allowed to initiated the commands to the Janitor, no further interaction from the site administrator is required. An exception may be to confirm the consistency of the system when the machine has crahsed and the Janitor may still find jobs assinged to runtime environments that are no longer running.

Another exception for an active involvement of the site administrator is the initial configuration of the Janitor and the updating of runtime environments that are eligible to be installed.

---

[*] Should you find that constraint unbearable for your purpose, please investigate the file rjanitor.cc in the ARC source tree. It wraps the janitor application and as a C binary can be configured to attract root privileges.

Table 3.1: Overview about the available commands to the Janitor.

janitor [COMMAND] [JOB-ID] [RTE] . . .	
Command:	
register	Registers a job and a set of runtime environments in the Janitor database. Requires the parameters [JOB-ID] and a list of [RTE]s.
deploy	Downloads and installs the desired runtime environments. Requires the name of an already registered [JOB-ID].
remove	Removes the placeholder of the job on the runtime environments. If there is no longer any job using the runtime environment and the lifespan of the runtime environment has be expired, the runtime environment can be removed using the `sweep` command. To indicate that a job has terminated, send the Janitor the remove command on that job's ID.
sweep	Removes unused runtime environments. No further arguements are required. Using the option `--force` enforces the removal of all unused runtime environments. Runtime environments having the state FAILED will not be removed.
setstate	Changes the state of a dynamically installed runtime environment. This might be useful in case a runtime environment with a state FAILED shall be removed (new state might be REMOVAL_PENDING). Requires the argument [STATE] followd by a list of [RTE]s.
search	Performs a simple search in the catalog and the manually installed runtime environments (`runtimedir`). Requires no [JOB-ID] nor [RTE]s, but only a list of string to be searched for.
list	Lists all information about jobs, automatically installed runtime environments and manually installed runtime environments. No additional parameters have to be passed.
info	Renders information about a job. Requires the parameter [JOB-ID].
Job id:	
	A unique sequence of numbers. Once the Janitor registered a job ID, it cannot register a second job having the same job id.
Runtime environments:	
	Runtime environments are defined by a continuous string. The name of valid runtime environment names can be investigated using the `list` or the `search` commands. They are defined in the catalog or by the directories and scripts of the `runtimedir` of the `grid-manager`.

Table 3.2: Possible exit states of the janitor application

Exit status:

The exit status of Janitor depends on the used command.

register	0	Registration was successful. No noteworthy occurrences.
	1	Registration was successful but some runtime environments aren't installed yet. Deploy is mandatory.
	2	An error occured.
deploy	0	Sucessfully initialized job.
	1	Can't provide requested runtime environments.
remove	0	Sucessfully removed job or no such job.
	1	Can't provide requested runtime environments.
sweep	0	Always returns this exit code.
setstate	0	Changing the state was successful.
	1	Can not change the state.
search	0	Search sucessfully finished.
list	0	Successfully retrieved information.
info	0	Successfully retrieved job information.
	1	No such job.
	2	Error while retrieving job information.

4 Maintenance

This chapter explains how to maintain the catalog and the Janitor itself. In order to administrate the catalog, it is absolutly recommended (but not required) to use the ontology editor Protègè. The first section will explain how this is done. The second section gives a detailed explanation how to create new packages for the Janitor. In the last section, a typical use case in maintaining Janitor will be demonstrated.

4.1 Catalog

The Catalog describes runtime environments and is either served through a web server or distributed together with the Janitor. It is specifed by a (Resource Description Framework) RDF file assigned to the Janitor using the tag `catlog` within the configuration file (see 2.1). The format of the RDF file is defined by an RDF schema file `knowarc.rdfs` which can be found along with an RDF example file `knowarc.rdf` in the janitor source directory http://svn.nordugrid.org/repos/nordugrid/arc1/trunk/src/services/janitor/resources/catalog/.

In order to adminstrate the catalog, the ontology editor Protègè should be used. Figure 4.1 shows the editor while the MetaPackage `APPS/BIO/JASPAR-CORE-1.0` of the example file has been selected. On the left side of the editor the class browser is placed. Three main classes are prepared: `MetaPackage`, `Note` and `Package`. The `Metapackage` is a general platform independent description of a `Package`. It can be understood as a reference to a functionality that should be implemented at the remote site. But the exact instruction on how to install it is not given. The "instruction-level" comes with instances of its subclass `Package`.

The `MetaPackage` is described by the subclasses of `Note`. The class `Note` has two subclasses: `BaseSystem` and `Tag` to describe the `MetaPackage`. The `BaseSystem` describes the Debian release a `Package` refers to (i.e. here etch or sid). The class `Tag` provides small keywords which can be assigned to `MetaPackages` such that they can be easier found. `TarPackage` and `DebianPackage` or currently the only subclasses of `Package`.

They are representing the necessary information (i.e. URL or Packagename) for the installation. To provide an overview on how the classes are interacting with each other the Tables 4.1,4.2, 4.3, 4.4 and 4.5 are pictured.

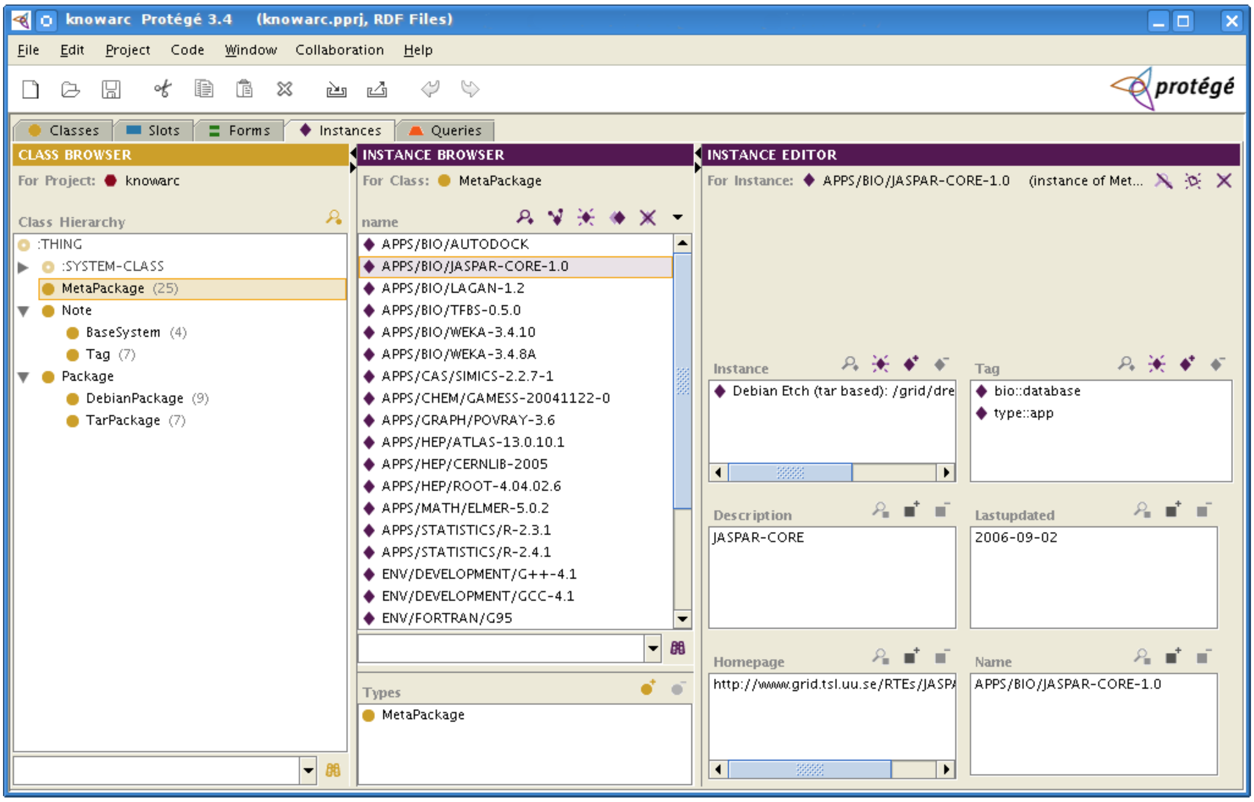
Table 4.1: Specification of class `Metapackage`.

Name	Cardinality	Type
description	single	String
homepage	single	String
instance	multiple	Instance of Package
lastupdated	single	String
name	required single	String
tag	multiple	Instance of Tag

4.1.1 Debian packages - dysfunctional in current implementation

The problem with Debian packages is that these are available only for the local machine and not immediately also for the whole network. This feature was meant for setups that use virtual machines for the execution of jobs.

Those entries can also be used to help the specification of further dependencies of runtime environments. It would then be left to the responsibility of the system administrator to manually (or assisted with scripts)



:THING
:SYSTEM-CLASS
MetaPackage (25)
Note
BaseSystem (4)
Tag (7)
Package
DebianPackage (9)
TarPackage (7)

APPS/BIO/AUTODOCK
APPS/BIO/JASPAR-CORE-1.0
APPS/BIO/LAGAN-1.2
APPS/BIO/TFBS-0.5.0
APPS/BIO/WEKA-3.4.10
APPS/BIO/WEKA-3.4.8A
APPS/CAS/SIMICS-2.2.7-1
APPS/CHEM/GAMESS-20041122-0
APPS/GRAPH/POVRAY-3.6
APPS/HEP/ATLAS-13.0.10.1
APPS/HEP/CERNLIB-2005
APPS/HEP/ROOT-4.04.02.6
APPS/MATH/ELMER-5.0.2
APPS/STATISTICS/R-2.3.1
APPS/STATISTICS/R-2.4.1
ENV/DEVELOPMENT/G++-4.1
ENV/DEVELOPMENT/GCC-4.1
ENV/FORTRAN/G95

Types: MetaPackage

Instance: Debian Etch (tar based): /grid/dre
Tag: bio::database
type::app
Description: JASPAR-CORE
Lastupdated: 2006-09-02
Homepage: http://www.grid.tsl.uu.se/RTEs/JASPA
Name: APPS/BIO/JASPAR-CORE-1.0

Figure 4.1: Example of a RDF catalog file as displayed in the program Protègè.

Table 4.2: Specification of class `BaseSystem`.

Name	Cardinality	Type
description	single	String
distribution	required single	String
name	required single	String
short_description	required single	String
url	required single	String

Table 4.3: Specification of class `Tag`.

Name	Cardinality	Type
description	single	String
name	required single	String

distribute a series of extra Debian packages throughout the compute nodes and use the catalog entry merely to indicate their presence.

One can as such interpret the catalog to represent an interface between software distributed via Linux distributions and independently from these via grid communities. It should be noted that the Linux distributions have now all started to accept communities to maintain packages, which may bringt many scientific packages away from being traditional ARC runtime environments towards becoming regular packages of some Linux distribution. For Debian, the Debian-Science and Debian-Med* communities are known to be very open to grid and cloud computing.

---

*http://debian-med.alioth.debian.org

Table 4.4: Specification of class `DebianPackage`.

Name	Cardinality	Type
basesystem	required single	Instance of BaseSystem
debconf	multiple	String
depends	multiple	Instance of MetaPackage or Package
package	required multiple	String

Table 4.5: Specification of the class `TarPackage`.

Name	Cardinality	Type
basesystem	required single	Instance of BaseSystem
depends	multiple	Instance of MetaPackage or Package
environ	multiple	String
url	required multiple	String

4.2 HTML interface of the catalog

The dynamic Runtime Environments stored in the catalog are presented on the aforementioned dedicated web page[†]. This site also links to both the formal Catalog in RDF syntax and an automated transformation to HTML. The latter mimics the traditional site describing Runtime Environments in the Runtime Environment Registry[‡] in order to minimise issues with an eventual transition to the new system.

That traditional page collects descriptions for Runtime Environments to encourage human site administrators to install these. This html page listing the manually or automatically installable RTEs is prepared by the script web/list.pl. This script is meant to be run by a mod-perl enabled Apache web server. The script itself is only loosely integrated with the remaining functionality of the Janitor, used only for the presentation to users. In the first lines of the script some variables specific to the site are set. To configure the script these have to be changed [**?**, p. 9].

4.3 Introducing new packages

This section describes how to add new packages to the catalog. Currently only tar based packages are processable by the Janitor. Within the here presented example the packages are assigned to be used together with Debian Etch. This limitation is only literal. There should be no limitiation to newer Debian distributions.

4.3.1 Debian Etch (tar based)

At the time of writing, only the tape archive (tar) file format is accepted for dynamic runtime environment installation. It is well accepted file format throughout the UNIX community. This section explains the inner structure of the tar files for the representation of dynamic runtime environments. Subdirectories are visualised in Figure 4.2.

The tar file contains two directories:

data/ contains all software that the grid-job may need

control/ contains files formally specifying how to deal with the information in the data/ directory.

While installing such tar-based runtime environments, the content of the data directory is extracted to some directory $BAR. After this unpacking of the tar file, the Janitor executes the install script provided in the

---

[†] http://dre.knowarc.eu:8080/list.pl

[‡] http://gridrer.csc.fi/

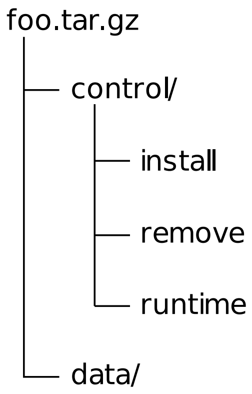


Figure 4.2: Directory structure in the tar files for automated installation.

control directory. It is executed within the working directory $BAR. The job of this skript is to perform any necessary post-processing. But the working directory shall not be moved. All post-processing needs to be performed *in situ*.

The Janitor stores the file `control/remove`. It will be executed in the same way as `control/install`, just before the tar-package is removed. In most cases `control/remove` will be empty, implying that the working directory $BAR shall be remove and no other action is required.

The remainder actions are regular actions performed upon execution of every grid-job. Upon submission, the file `control/runtime` is sourced multiple times by the Grid Manager's job-submit script. Every ARC runtime environment must specify such a runtime script, new is only its specific location as control/runtime. Since the directory $BAR is not known for the individual preparing the runtime environment, that file will instead use the placeholder %BASEDIR%. After installing the package, the Janitor changes all occurences of %BASEDIR% in the runtime script to $BAR. To be offered to computing elements for an installation, the such prepared runtime environment must be announced to a Catalog to which the Janitor on the computing element subscribes [**?**, p. 10].

4.3.2 Example: WEKA machine learning Java library

The WEKA package for machine learning [**?**] and the Java Runtime Environment are made available as dynamic Runtime Environments. Further packages for bioinformatics comprise dynamic variants of tools for the analysis of transcription factor binding sites. These are already offered for manual installation via the prior mentioned traditional page representing Runtime Environments for ARC. The corresponding tar file is named somewhat??. The `data` directory simply contains a ZIP file which needs to be unzipped in the installation directory. For that reason, the `control/install` script is written as follows:

```
#!/bin/sh
set -e  # Makes the script to terminate at the first line it fails.

WEKA_ZIP="weka-3-4-8a.zip"
unzip $WEKA_ZIP
rm -f $WEKA_ZIP
```

The runtime script sets the environment variable of the Java Classpath:

```
#!/bin/sh

WEKA_JAR="weka-3-4-8a/weka.jar"
case "\$1" in
0) # Just before job submission
# none
```

```
;;
1) # Just before job execution
# Initialize the java environment
CLASSPATH="%BASEDIR%/$WEKA_JAR:$CLASSPATH"
export CLASSPATH
;;
2) # After job termination
# none
;;
*)
return 1
;;
esac
```

The remove script, which will be executed right before WEKA is deinstalled, is empty. Janitor will delete the directory, so there is nothing to be done.

4.3.3 Example: R packages in CRAN and BioConductor.org

To demonstrate the technical proximity to scientific communities that provide packages for the Debian Linux distribution, a tool was prepared to transform Debian packages to dynamic runtime environments[**?**]. This effort comprising more than 1700 packages and thus also helps to analyse the scalability of the RDF-based tool for the analysis of dependencies between projects.

This effort has not found practical applications beyond this proof of concept. R packages are now frequently found in regular Linux distributions. And for most packages, it is reasonably well possible to install the packages at runtime.

4.3.4 Example: ATLAS for High Energy Physics

To address the concerns of the physicists using ARC, a dynamic runtime environment for the ATLAS software suite was prepared. It extends prior work on an automated installation that is available at http://guts.uio.no/atlas/12.0.6/. The preparation comprised the following steps:

- The file system path specifications in the automated installation scripts were modified using the Janitor path variables.
- A tarball was prepared containing a directory structure as illustrated in Figure 4.2. The data directory was empty, since the automatic installation script downloads the software from a remote server.
- An entry was added to the Catalog file.

What sets High Energy Physics software apart is it's sheer size. The package in question takes up more than 5 GB. This was a test illustrating the feasibility of using DREs in High Energy Physics. The application of the DRE for ATLAS needs to wait for the planned web service extension of the Catalog. With such a service, e.g. a software manager of a big experiment will be able to deploy software packages on production sites simply by creating a tarball and adding an entry to the Catalog.

4.3.5 Debian Etch

These kind of packages are not yet supported.

4.3.6 Debian Sid

These kind of packages are not yet supported.

4.4 Adminstrating the Catalog

5 Programming concept

The main language for the implementation of the functionality of the dynamic runtime environment functionality is Perl. And it is solely required (exceptions are the integration with the Grid Manager and the Web service) for the Janitor. In the pre-web-service implementation the Catalog remains a static web page. The Perl code is split into multiple modules as depicted in Figure **??**. The modules can be separated into two functional groups. One addresses the retrieval of information from the Catalog RDF file in the left major branch of the figure. The other addresses the process of fetching and installing the packages.

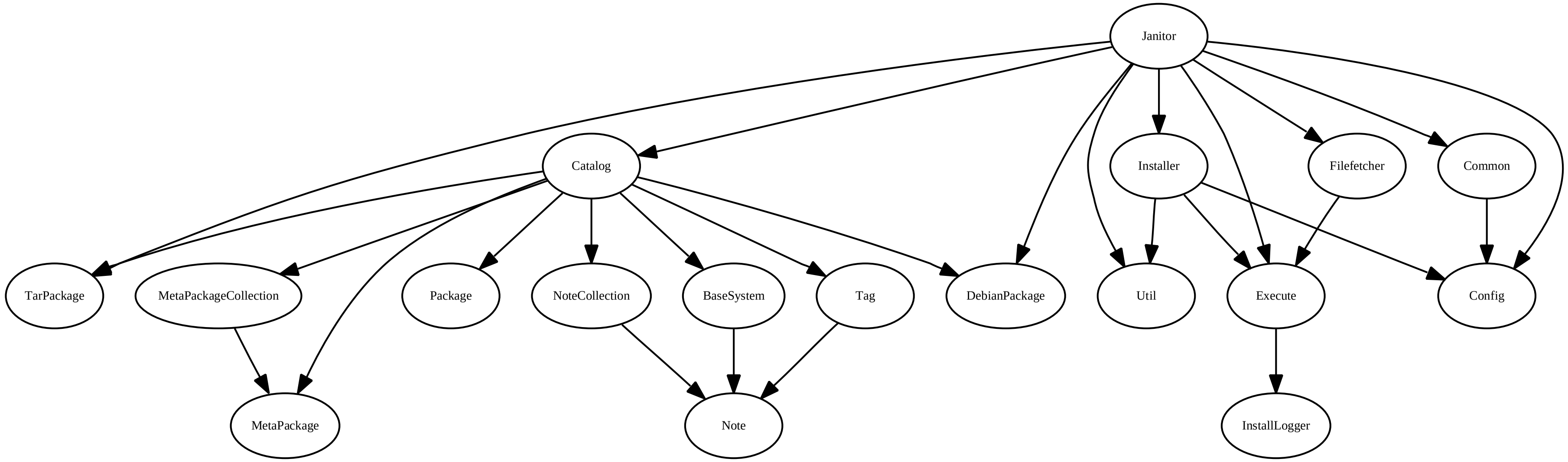


Figure 5.1: Modules of the Janitor and their dependencies

In order to get a more detailed view on the full functionality of the envisioned system it is suggested to consult the Design Document*.

5.1 Runtime environments states

A main motivation for the managed, manual initiation of dynamic RE installation is the subsequent manual verification of the installed packages – prior to their use in production.

With an automation of the installation, the verification of that process shall be performed externally to that process. At this time, only the automation of the installation has been implemented. To reflect the progress the external verification has made, REs are said to be in states. The current implementation lists installable REs aside the installed REs in the grid information system, in order to stimulate grid clients to submit packages. The here described states will be represented to the clients in upcoming developments.

These states are specific for every CE and communicated between the Janitor and the execution service A-REX . Table 5.1 shows all possible states, while Figure 5.2 displays the transitions between the states that a Runtime Environment may be in during its life time.

State	Description
UNAVAILABLE	The RE is not available for the BaseSystem (see 4.1) the site uses.
INSTALLABLE	The RE is available for the BaseSystem the site uses and it will be automatically installed once a job requests it.
INSTALLING/a	A job requested the RE and it is currently being installed
INSTALLING/m	The RE-adminstrator requested the installation of the RE. Its currently being installed.
FAILED	The installation process failed.
INSTALLED/a	The RE is installed dynamically.
INSTALLED/m	The RE ist installed manually by the RE-administrator
BROKEN/m	The RE is installed but failed tests of the RE-administrator
VALIDATED/m	The RE is installed and successfully passed the tests of the RE-administrator
REMOVAL PENDING	The RE is still installed but will be removed as soon as possible. It is not available to new jobs.
REMOVING	The RE is currently being removed.
INSTALLED/s	The RE was installed in the traditional way by the site administrator.
BROKEN/s	The RE was installed in the traditional way and failed validation by the RE-administrator,
VALIDATED/s	The RE was installed in the traditional way and was successfully verified.

Table 5.1: States a Runtime Environment can possibly be in.

The concept of ARC prevails to have a single directory into which to install software to be execute on all compute elements of the site. Otherwise, the installation of a runtime environment would be required to be performed just when the jobs initiates its computation.

In principle this would be doable, particularly for those runtime environments that are part of a trusted Linux distribution already. But this would also mean that the compute nodes become inhomogeneous, and at the time of writing this was considered undesireable.

The manually induced transitions are marked in red, he automated transitions in black. A transition between states can be induced automatically (i. e. by the advent of a job requesting a particular dynamic RE) or manually by the site's administrator or an individual with respective rights to use the Janitor's command line.

---

*http://www.knowarc.eu/documents/Knowarc_D1.1-1_07.pdf

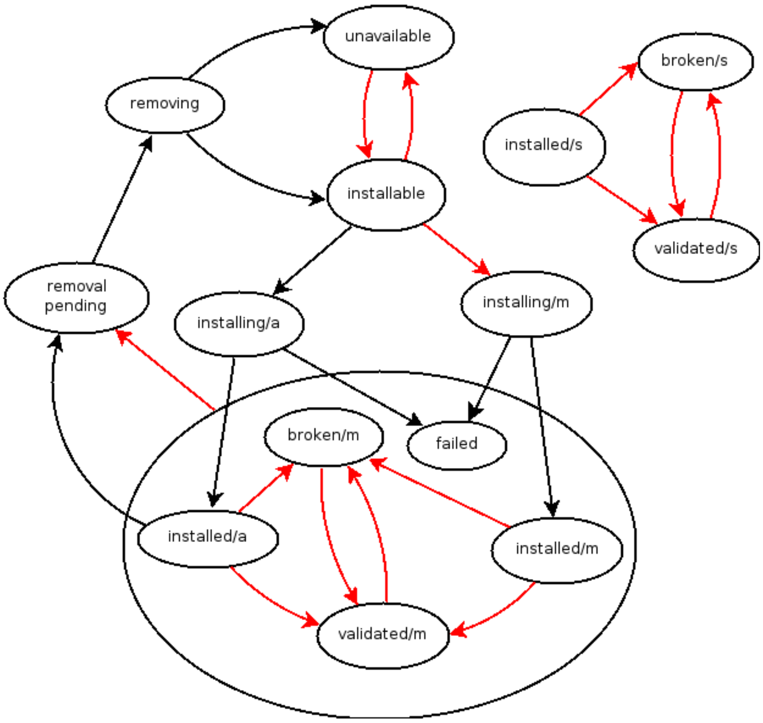


Figure 5.2: Relationships between the possible states of Runtime Environments. Red arcs represent human interaction. The distinction between **/a**, **/m** and **/s** states does not need to be visible for all clients.

Upon presentation of a package to a Catalog, a CE may classify a package to be `INSTALLABLE` if all the dependencies are installable or already `INSTALLED`. The installation can be performed manually (`INSTALLING`/**m**) or in an automated fashion (. . . /**a**). Should the installation process return an error, then the installation has `FAILED`. Once the installation succeeded, the installed package is validated for its correctness. Should that process fail, then the package's state it is said to be `BROKEN`.

Automatically installed packages can be removed by the automatism. A manually installed package or one that has failed to be installed, can only be removed upon manual induction. The . . . /**s** states represent those Runtime Environments that are installed in the original manual way of RE installation in ARC 0.6.

5.1.1 Dependencies on Job IDs

The Janitor was designed with the job execution in mind. As such, all actions it performs are driven by the need of a job. And those jobs should then be specified, to learn when the demand for a particular runtime environment has ended.

This concept partially conflicts with the idea to use the Janitor as an aid for the manual invocation of an installation – there is no job ID that could be assigned for that process. Today, the administrator is suggested to use a special number, e. g. 0, to indicate such manual installs. This will change in near future.

5.2 Job states

The Janitor has two states for jobs: `PREPARED` and `INITIALIZED`. After a job has been succesfully registered with the Janitor, its state will be set to `PREPARED`. Invalid jobs are not cached by the Janitor. After the Janitor is requested to deploy the runtime environment, the state of the job will change to `INITIALIZED`. If an unforeseen exception occures during that process, Janitor will drop the job from its database and set the affected runtime environments to the state `FAILED`.

5.3 Integration into AREX

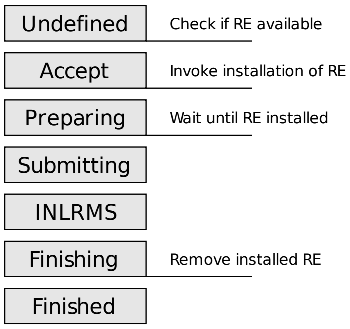


Figure 5.3:

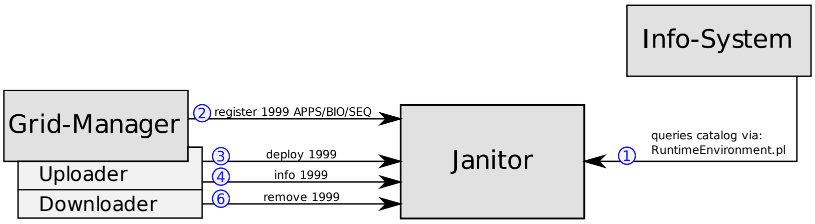


Figure 5.4:

5.4 WebService Interface

Default port number: 55555
Client command equal, except assignment of HED.xml
(from /arc1/trunk:12561)

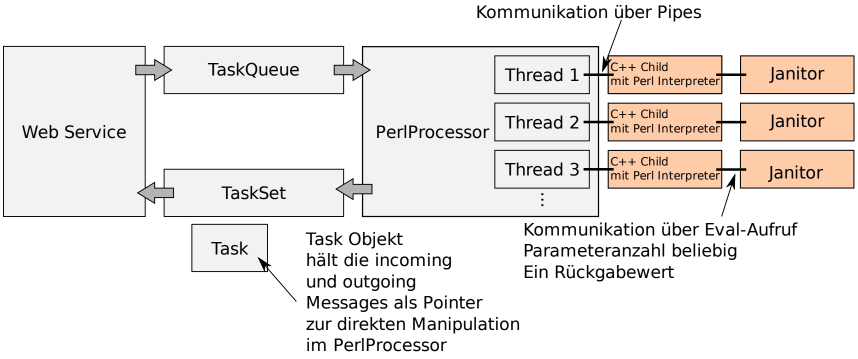

Kommunikation über Pipes
Web Service
TaskQueue
TaskSet
PerlProcessor
Thread 1
Thread 2
Thread 3
C++ Child mit Perl Interpreter
C++ Child mit Perl Interpreter
C++ Child mit Perl Interpreter
Janitor
Janitor
Janitor
Task
Task Objekt hält die incoming und outgoing Messages als Pointer zur direkten Manipulation im PerlProcessor
Kommunikation über Eval-Aufruf
Parameteranzahl beliebig
Ein Rückgabewert

Figure 5.5: To be translated and beautificated. SVG file is missing!

Proposal for SOAP messages:

namespace: dynamicruntime or janitor
Create WSDL files for that Permission concepts: Depending on certificates. Certain certificates may `sweep`. Defined in service_HED.xml. Evaluated in:??

Listing 5.1: Example *arc.conf* settings for janitor.

```
<Request action="SEARCH|SWEEP|LIST|DEPLOY|REMOVE|CHECK|REGISTER">
    <Initiator jobid="1234"/>  <!-- Needed for: CHECK|REGISTER|DEPLOY|REMOVE -->
                                  <!-- May contain no jobID in this case a new one will
                                       be created and returned via the response message-->

    <Runtimeenvironment type="dynamic">      <!-- Needed for: SEARCH|REGISTER-->
        <Package name="APPS/BIO/WEKA-3.4.10"/>
        <Package name="APPS/BIO/WEKA-3.4.11"/>
    </Runtimeenvironment>
                                  <!-- SWEEP and LIST only works, if the TLS-adminstrator identity
                                       (which is assigned in the arched configuration file) is
                                       to be found by the SecHandler. Both need neither initiator
                                       nor runtimeenvironment elements -->
</Request>
```

Listing 5.2: Example *arc.conf* settings for janitor.

```
<response action="SEARCH|SWEEP|LIST|DEPLOY|REMOVE|CHECK|REGISTER">
    <initiator jobid="1234"/>  <!-- Needed for REMOVE|REGISTER|DEPLOY|CHECK-->
    <result code="0" message="Sucessfully initailized job."> <!-- -->
    <jobs> <!--LIST|CHECK-->
        <job jobid="1234">
            <created>1234567890</created>    <!-- in unix time-->
            <age>0</age>           <!-- in seconds-->
            <runtimeenvironment>
                <package>APPS/BIO/WEKA-3.4.10</package>
            </runtimeenvironment>
            <state>INITIALIZED</state>
        </job>
        <job jobid="4321">
            <created>1234567891</created>
            <age>0</age>
            <package>APPS/BIO/WEKA-3.4.10</package>
            <state>INITIALIZED</state>
            <runtimeenvironmentkey>APPS_BIO_WEKA_3_4_10-835614b62c98c4eb6cb03d74d3161b5d</runtimeenvironmentkey> <!-- at least CHECK -->
            <uses>/nfshome/knowarc/dredesign/src/services/dRE3/perl/spool/runtime/jre__57T1ke1UVz/runtime</uses> <!-- at least CHECK -->
            <uses>/nfshome/knowarc/dredesign/src/services/dRE3/perl/spool/runtime/weka_wHfyytarlE/runtime</uses> <!-- at least CHECK -->
        </job>
    </jobs>

    <runtimeenvironment type="local">      <!-- Needed for: LIST|SEARCH -->
        <package name="APPS/BIO/MUSTANG-3.0-1"/>
        <package name="APPS/BIO/EXONERATE-2.1.0-1"/>
    </runtimeenvironment>

    <runtimeenvironment type="dynamic">      <!-- Needed for: LIST -->
        <package name="APPS/BIO/WEKA-3.4.11">
            <state>INSTALLED_A</state>
            <lastused>1234567890</lastused>
            <jobid>1234</jobid>
        </package>
        <package name="APPS/BIO/WEKA-3.4.10">
            <state>INSTALLED_A</state>
            <lastused>1234567890</lastused>
            <jobid>1234</jobid>
            <jobid>4321</jobid>
        </package>
    </runtimeenvironment>

    <runtimeenvironment type="installable">      <!-- Needed for: LIST -->
        <package name="APPS/GRAPH/POVRAY-3.6">
            <description>The Persistence of Vision Raytracer</description>
            <lastupdate>1234567890</lastupdate>
        </package>
        <package name="APPS/BIO/WEKA-3.4.8A">
            <description>WEKA Machine Learning Software</description>
            <lastupdate>1234567890</lastupdate>
        </package>
    </runtimeenvironment>

</response>
```

5.5 Janitor file system permissions

At the time of writing, the Janitor needs to be executed as root. This may change over time, but as long as all other components of ARC also demand superuser privileges, one can expect the demand for the Janitor to be non-problematic.

5.5.1 Security Consideration

Security is a major concern for the grid systems. An automatic software installation inheritently introduces security threats. This section addresses those and describes the available solutions to limit security risks.

In the current installation, every user authorised to execute a job is also authorised to install a REs. Restrictions are only imposed on the set of DRE that are available for installation. Restrictions are imposed by the site admininistrators on the descriptions that are given by the Catalog that is offering the package. These descriptions may explicitly mention DRE names, a regular expression on these, or refer to tags of packages that categorise these. However, the core of these controls lie with the maintainers of the Catalog, who needs to be trusted.

All dynamic REs are installed in separate directories. The provisioning of disk space is the duty of the site administrator. In the current implementation, the installation is completely transparent to the user:

- DREs are not distinguished between *installed* and *installable* in the information system.
- No status information is given at the time an DRE is installed.

Malevolent regular users with respective training in using system exploits to gain root access are likely to find security holes by regularly submitted scripts. The authentication and authentification of users, together with respective logging, is the major defense against such attacks. What is consequently left to be protected against are unwanted side-effects by the installation of software.

The worst case scenario would be the installation of a RE that overwrites system files. With the current implementation, which is based solely on tar files, this is barely possible, unless such is performed by the install scripts that accompany the tar files.

The installation of packages from the Debian distribution (or other packages of mainstream Linux distributions) is seeked to reduce the complexity and burden in the maintenance for DRE. In the current implementation Debian packages may be installed only by their transformation into tar files. With the advent of the interface for the virtualisation of the grid infrastructure, it is anticipated to work with native packages of the Debian Linux distribution. The reuse of packages that passed many eyeballs - as it is the case with packages from major Linux distribution - security is further increased or becomes as high as with the operating system underneath virtual clients.

As a conclusion, there is general concern about the security of grid computing. Dynamic REs introduce new dangers since a manual control at the grid site is substituted by a remote process that is out the direct supervision of a local site administrator. The signing of packages by known and directly or indirectly trusted developers is a good indicator that no malevolent individuals have tampered with the binary. The site administrators can limit the sources of packages and specify packages that are eligible or excluded from installations.

6 Outlook

6.1 Representation of dynamic REs in the information model

Dynamic REs require an extended representation in the information model. The Application Software description should be able to distinguish installed REs from installable REs, potentially offer descriptions of extended RE state-like information. This work is planned to be carried out as part of the Glue-2.0 effort of OGF*.

6.2 Integration with Workflow Management

Future development of ARC aims at integrating grid computing with workflow tools for the web services that have a growing user base in bioinformatics. The challenge is to prepare REs for programs or databases and to offer such concisely to users of the workflow environments. In the bioinformatics community, such are today offered as web services. This anticipated development instead fosters the dynamic installation on the grid whenever appropriate to allow for special computational demands in high-throughput analyses. Conversely, because of the increased complexity of workflows with respect to the already today not manually manageable number of REs, without an automatism for the automated installation of software packages on the grid, the use of workflows in grid computing seems mute.

6.3 Implementation of a Catalog service

A Catalog service is planed to be implemented on top of the ARC HED component. This service will render the currently used locally accessible RDF file externally accessible. Selected users are then allowed to remotely add/edit/remove REs to/from to it. The Janitor will access the content of the Catalog through a well-defined Web Service interface.

6.4 Integration with the Virtualization work

The RDF schema nicely prepares for the upcoming virtualisation of worker nodes. How exactly the dynamics are integrated will depend on how dynamic the virtualisation of the nodes is. In the simplest scenario, a worker node's CPU will only be occupied by a single virtual machine and that will not be changed. In this case, there is no difference to the setup of the Janitor with today's static setups.

However, if the BaseSystems can be substituted dynamically, then a RE can possibly be offered via multiple BaseSystems. The RDF Schema describes BaseSystems as separate instances and as such differs from the current RE registry. Heuristics that prefer one BaseSystem for another can make direct use of the data that is presented in the schema. The integration of packages from Linux distributions in the description of REs is essential to have a means to decide for the equivalence of manual additions and the functionality that comes with BaseSystem.

6.5 Use of RDF

The RDF is "correctly" used within the Janitor via the Redland libraries. What is still missing is a semantical reasoning on what packages to allow or disallow. For instance: allow all packages associated with

---

*OGF GLUE: https://forge.gridforum.org/sf/projects/glue-wg bioinformatics. This shall wait a bit longer until more experiences on the community's demand for this service have been made.

6.6 Manual Verification

There is yet no explicit notion of a concept on how users can report on the reliability of individual compute nodes and use such information for the decision making on where to sent their jobs. And with an increasing complexity of software installed, being used across versions, one will rather trust one's very own experiences for individual sites than some external repository or other pieces of information that may be weeks or months old.

Consequently, no means for a manual verification and the communication of results of such have yet been implemented.

6.7 Known issues

- There is a arc.conf file in which all possiblie flags are listed... ADD THE JANITOR FLAGS!!!
- Dynamic RTEs are now listed as manually installed in the "janitor list" command.. change that

7 Appendix

7.1 Useful tutorials and documentations

- **Another document describing Janitor.**
  D2.5-1 RDF Based Semantic Runtime Environment (RE) Description And Dynamic RE Management Framework Including Creating Proof Of Concept Bioinformatics REs, Daniel Bayer and Steffen Mller and Frederik Orellana[**?**]