Enabling Grids for E-science in Europe

www.eu-egee.org

ARDA 21/10/04

Package Manager

Predrag Buncic
JRA1

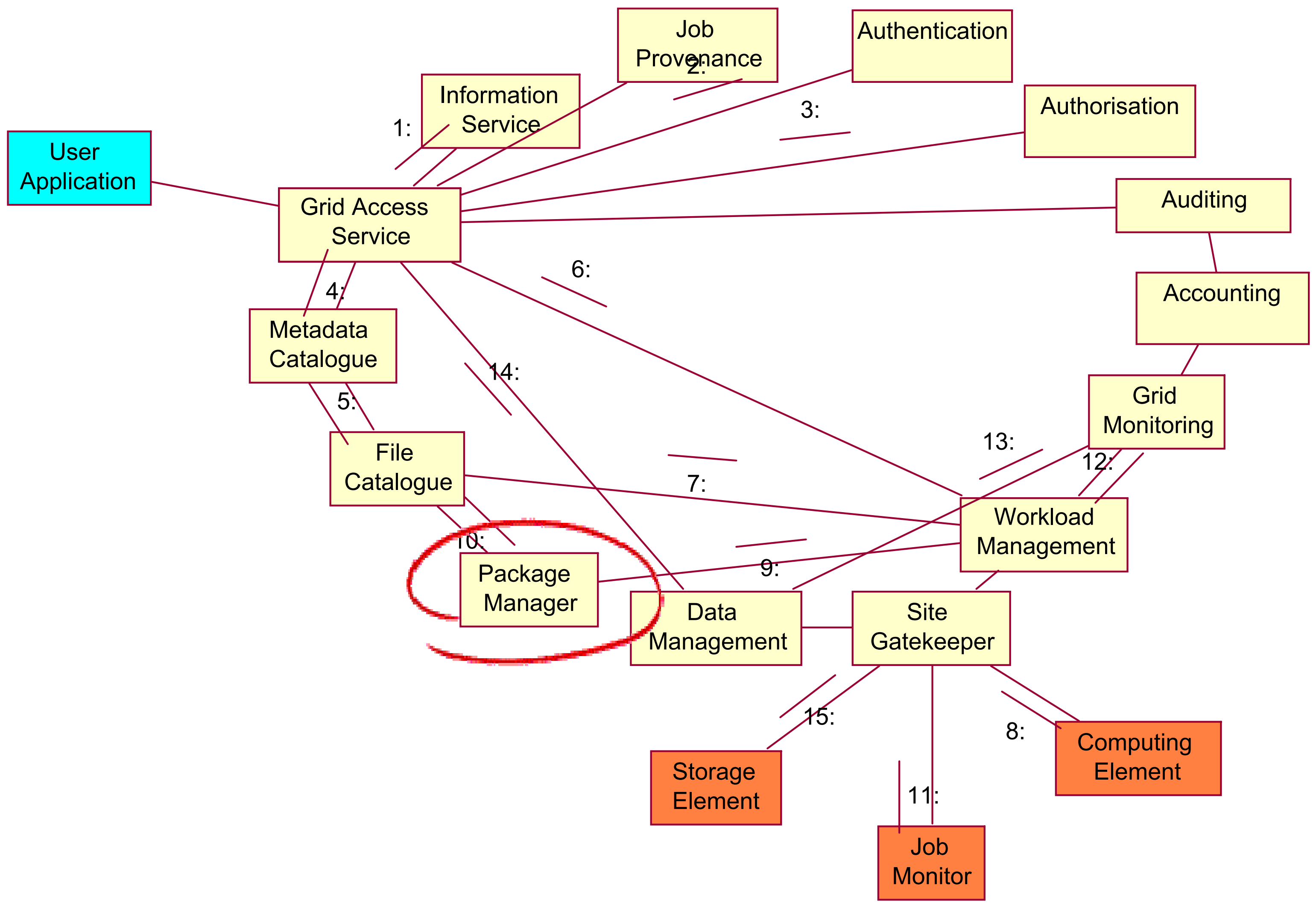
EGEE is a project funded by the European Union under contract IST-2003-508833

Talk Outline

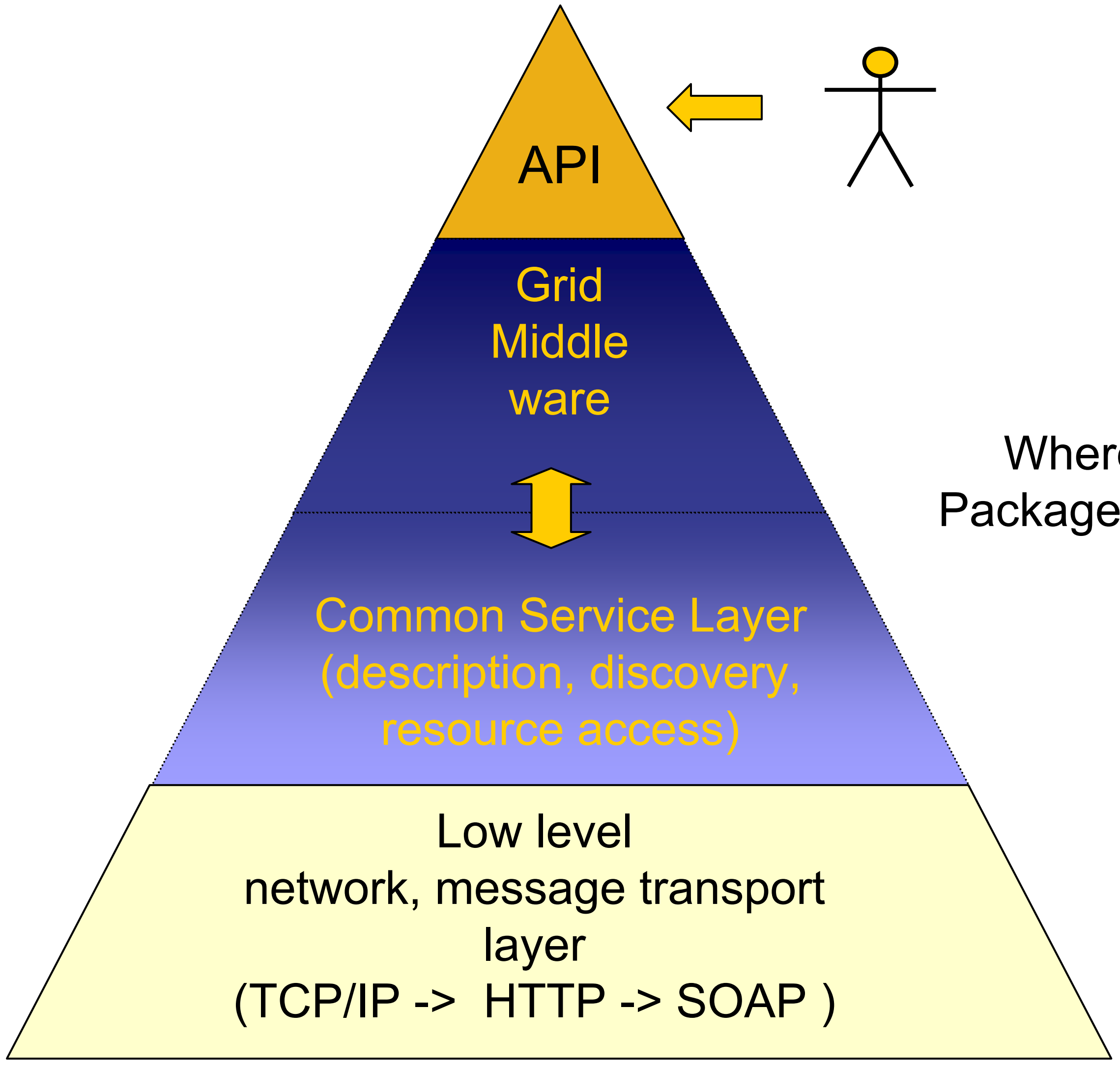
Enabling Grids for E-science in Europe

- Introduction
  - gLite services and Package Manager
- Design
  - Interface
  - Use Cases
- Current prototype
- Outlook & Conclusions

ARDA Service decomposition

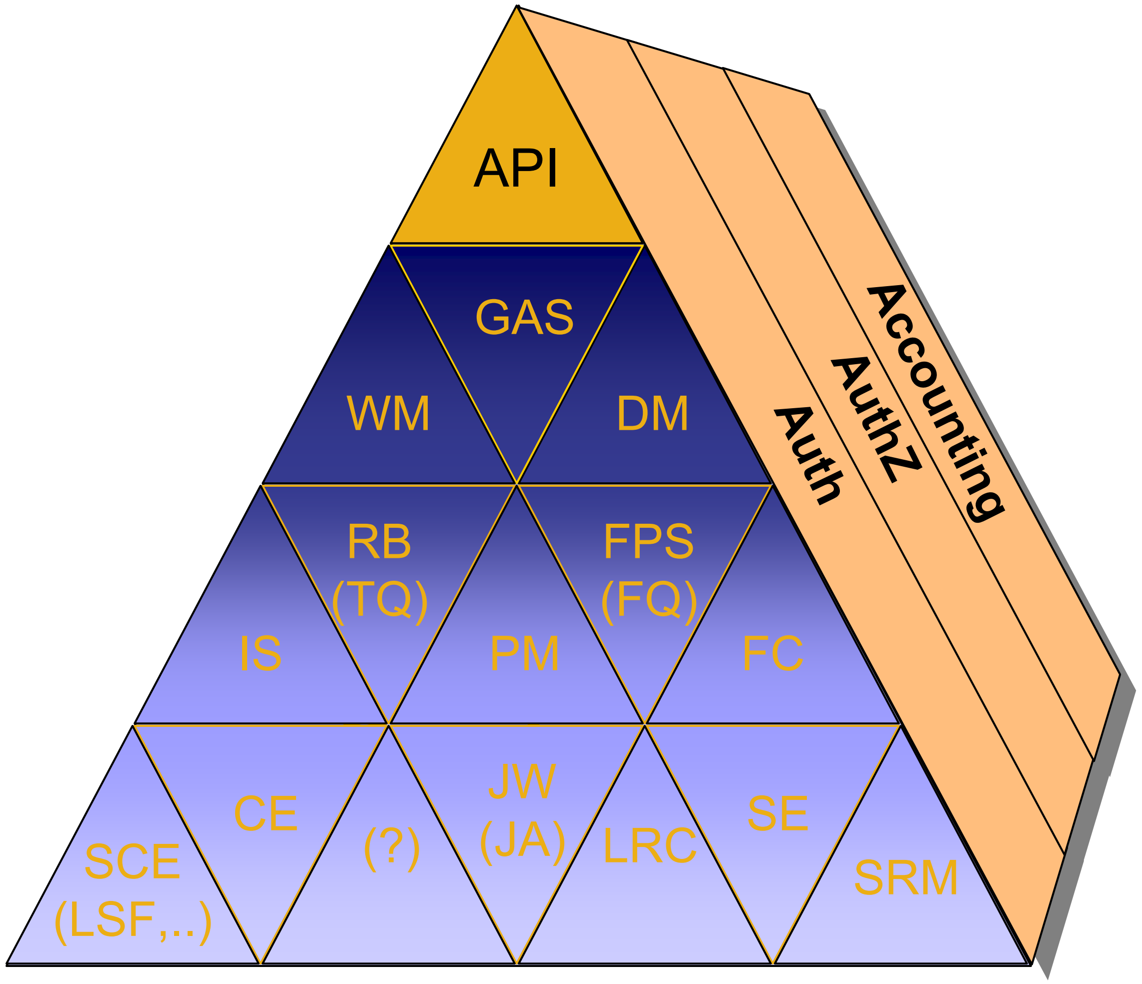


Grid Middleware


API
Grid Middle ware
Common Service Layer (description, discovery, resource access)
Low level network, message transport layer (TCP/IP -> HTTP -> SOAP)

Where is the Package Manager?

Middleware Services in gLite



"Periodic System of Grid Middleware Services"

Package Manager Service

- This service is a helper service that automates the process of installing, upgrading, configuring, and removing software packages from a shared area (software cache) on a Grid site

- The Package Manager Service does not pretend solve the problem of sharing a software cache between worker nodes

  - There are many possible deployment scenarios and solutions which are ultimately matter of choice and responsibility of a site managers
    - Shared file system (AFS, NFS, ..)
    - Another service dedicated to this purpose

- The Package Manager Service does not manage the installation of middleware software

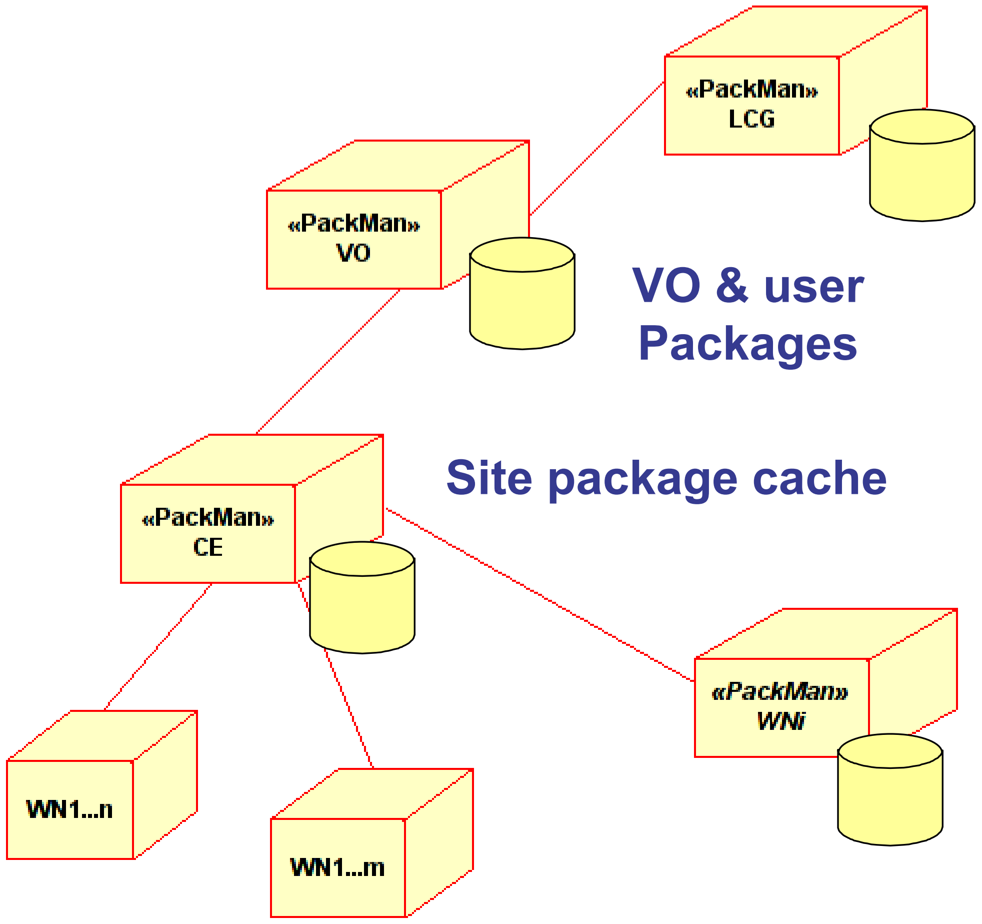
Assumptions & Constraints

- The software is distributed in packages, usually encapsulated into a single file that contains payload and the metadata that describes the package's details
  - Name and checksums
  - Dependencies on any other packages that it needs to work
  - Instructions how to setup the execution environment
  - Information on how to remove the package cleanly when it is no longer required
  - Pre and post installation scripts (including verification)
- The packages are installed on demand, when requested by the
  - Job Agent/Wrapper running on a worker node or another service
  - VO Software Manager
- The Package Manager Services can form a hierarchy
  - WN => CE => Site => VO => Super VO
- Packages can be defined either by a VO or by a user
- No root access

Interface

- *packageAdd(user, packageName, version, url)**
  - Registers the package *packageName* of a given *version* that belongs to *user* and can be found at *url*
- *packageInstall(user, packageName, version, TTL)*
  - Installs and verifies the package *packageName* of a given *version* (if not already installed) and extends a lease time for *TTL* (time-to-live) hours in the name of *user* (or *service@host*, *job@host*)
- *listPackages()*
  - Returns the list of packages currently installed in the cache of a package manager instance
- *removePackage(user, packageName, version)*
  - Unconditionally removes the package *packageName* of a given *version* from the package manager cache

(Possible) deployment scenarios



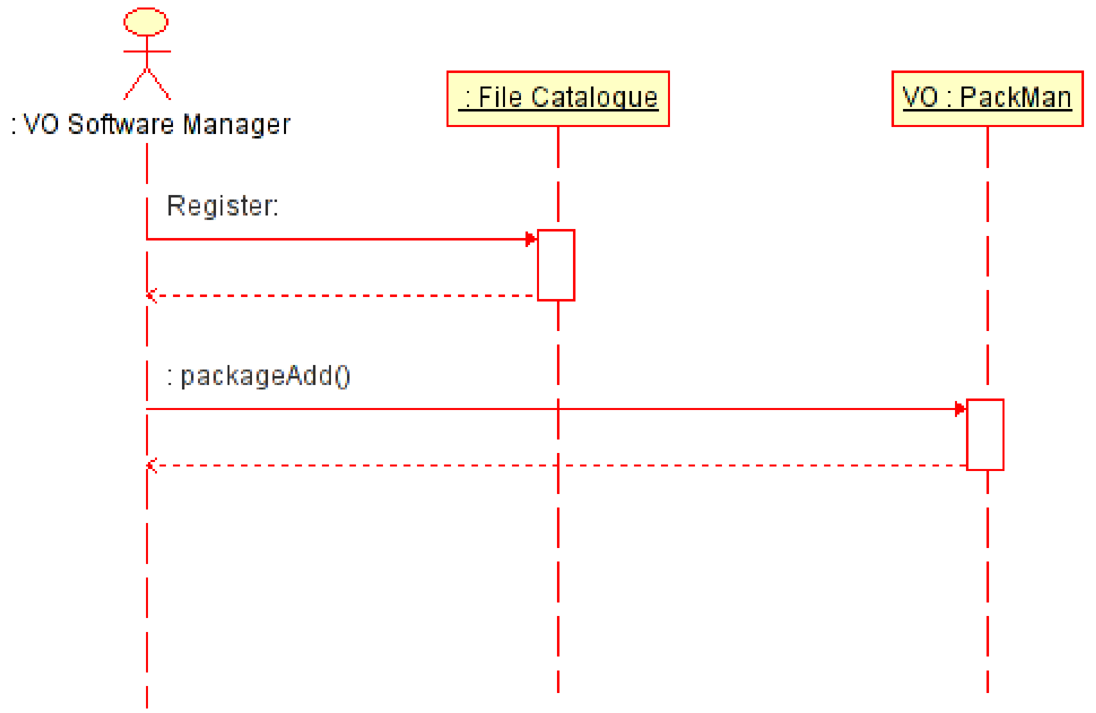
Common Packages (ROOT, POOL,..)

VO & user Packages

Site package cache

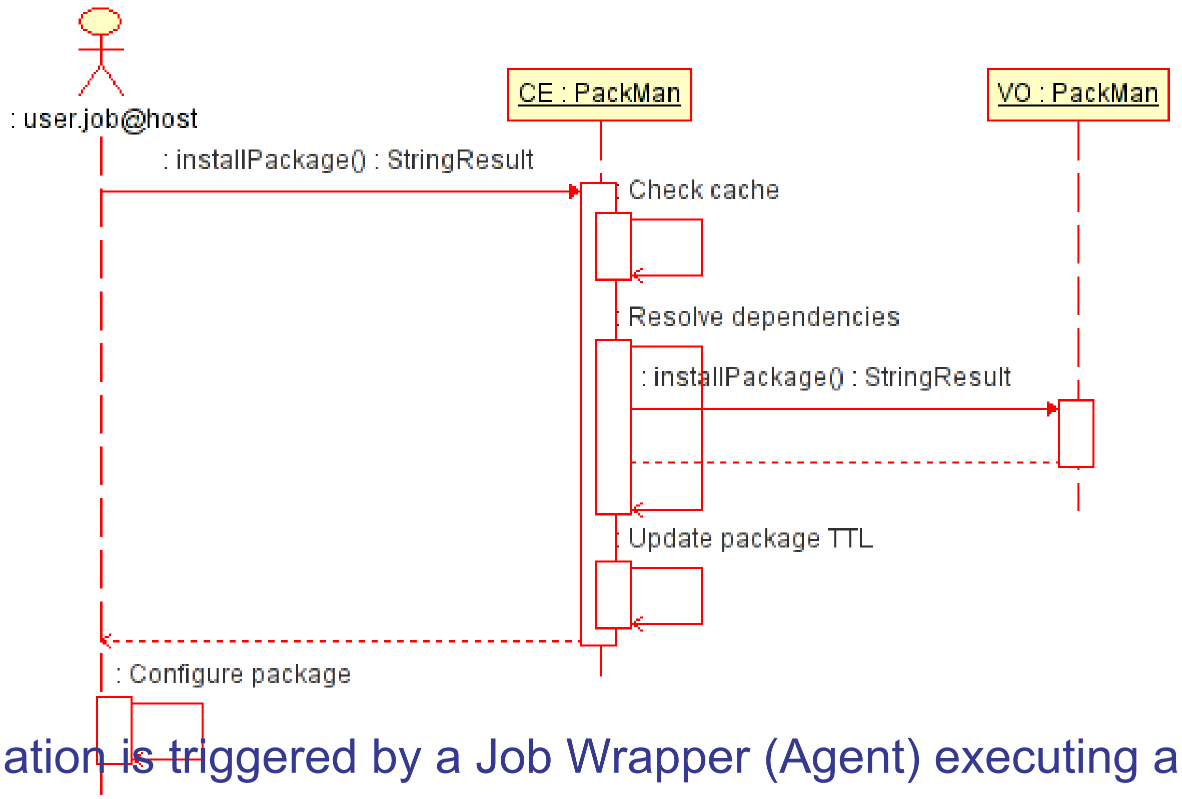
Worker node cache

Package creation



- ✓ VO Software Mgr (or a user) can register a package
  1. VO package manager (or a user) creates the binary package for one or more platforms and registers the content and metadata in the file catalogue
  2. Package is registered with VO Package Manager Service
     - This is not yet implemented in the prototype

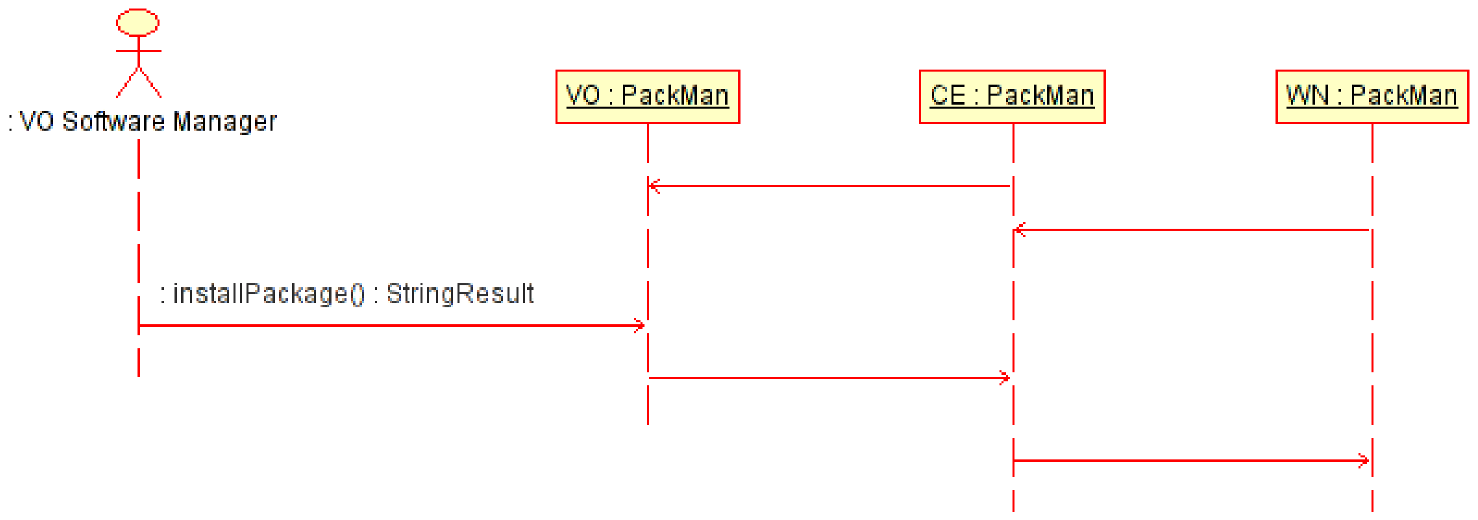
Installation on demand



✓ The installation is triggered by a Job Wrapper (Agent) executing a job on WN

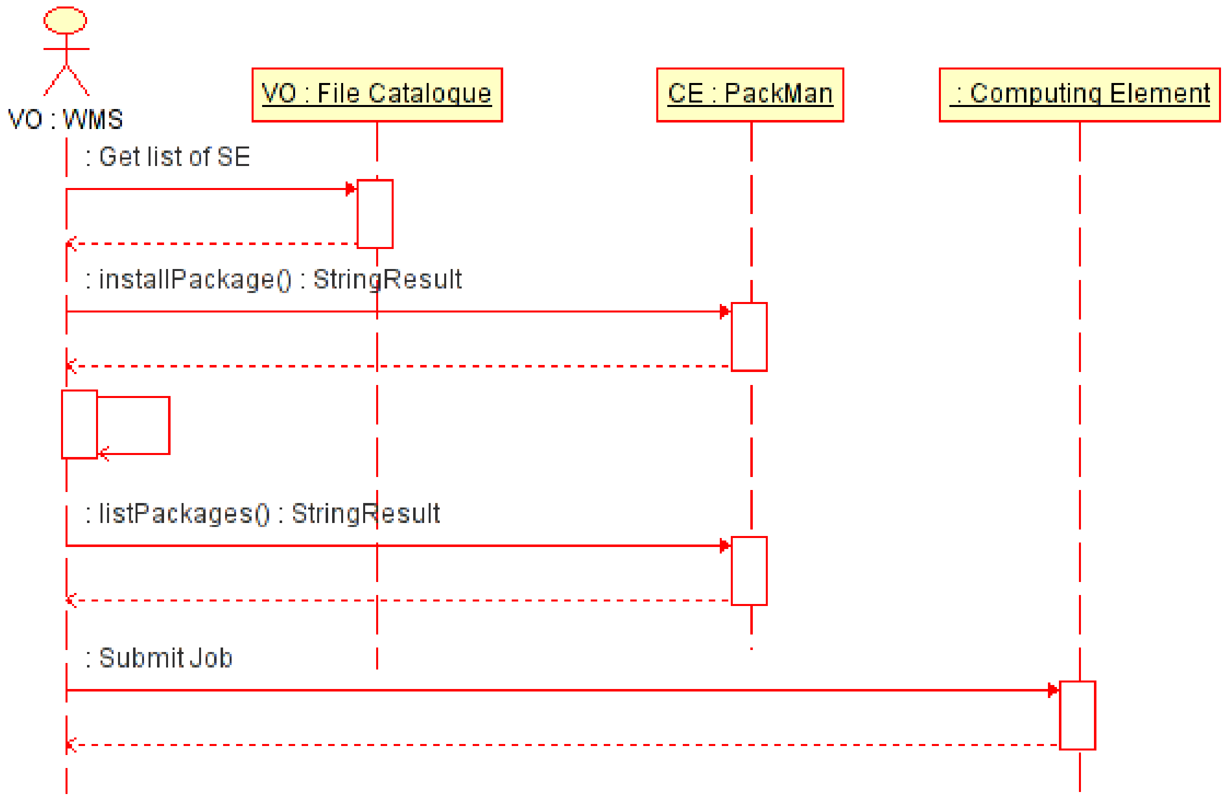
1. JW requests a package from local PM
2. PM checks the cache and if does not find a package, asks parent manager
3. Local PM installs all dependencies in the cache
4. JW obtains a lease for the package (PM extends TTL)
5. Job Agent receives an instruction how to setup the environment before executing a job

Pre-installation



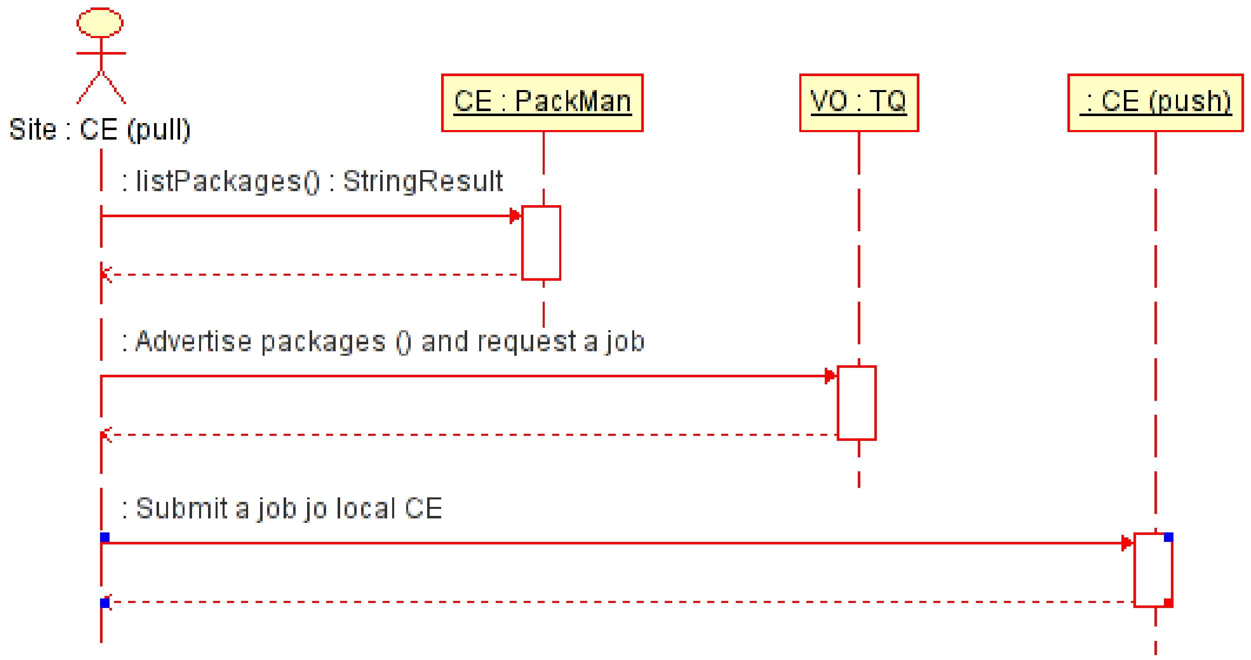
- ✓ In this case the installation is triggered by the VO Software Manager
  1. Site PM registers with VO PM upon startup
  2. Any other PM service on the site registers with Site PM
  3. VO Software Manages issues installPackage command
  4. This is propagated down the hierarchy and appropriate TTL is set

WMS



- ✓ Workload Management can trigger software installation to optimize job execution
  1. WMS consults FC (and other services) to find out about possible sites for job execution
  2. It issues installPackage() command on PM associated with given CE and obtains the lease for the package
  3. It pushes jobs on the sites on which listPackages() confirms that package exists

Task Queue


Site : CE (pull)
CE : PackMan
VO : TQ
: CE (push)
: listPackages() : StringResult
: Advertise packages () and request a job
: Submit a job jo local CE

✓ The CE running in the pull mode

1. CE periodically monitors local resources and lists the packages available at its PM
2. The packages are advertised together with other parameters in a JDL which is presented to Task Queue
3. If CE is given a task, it builds a Job Wrapper (Agent) and sends it to the local CE (push)

Cleanup

- The Package Manager manages the local disk cache and will clear the disk space only if it needs the disk space to install new packages
  - It won’t remove the packages for which someone holds the lease
  - The maximum lease time for the packages is a configurable parameter
- While any user or process can list already installed packages, only the VO administrator can remove a package from the local cache regardless of its current lease status
- Removing a package does not remove the packages that depend on it
- If any of removed packages are requested again, they will be automatically installed again

Creating a package in gLite

- At present, we do not publish packages and do not implement Package Manager hierarchy
- We are using File Catalogue to deliver package and metadata content
- Creating the package

1. Creating tar file

```
tar czf ROOT.tar.gz
```

1. Registering the package in the catalogue:

```
(from the glite prompt)
    mkdir ~/packages/ROOT/4.0.8
    add  ~/packages/ROOT/4.0.8/Linux-i686 file://myhost/ROOT.tar
```

2. In the JDL of a job, require the package

```
Executable="myExec.";
Packages="ROOT:4.0.8";
InputFile=….
```

Additional package info

- It is possible to define additional package metadata
    - Size
    - Dependencies
    - configuration script
    - pre- and post-installation scripts
    - installation script
    - pre- and post- remove scripts

- To define any metadata the user has to :

    - Create the metadata structure for that directory

      ```
      addTag ~/packages/AliROOT PackageDef
      ```

    - Populate the metadata

      ```
      addTagValue ~/packages/AliROOT/4.0.2 PackageDef Dependencies=‘ROOT:4.0.8’
      ```

Possible evolution

Enabling Grids for E-science in Europe

- The PM Service should be part of the hierarchy of package managers to assure scalability and provide a fail-over capability.
- Access to VO packages should be controlled and possibly restricted and audited
  - To some extent this could be achieved by running “public” and “protected” instances of the service
- The package metadata information (including checksum information) should be digitally signed
  - The metadata should come from the database and be digitally signed while payload could be replicated
- The package metadata could contain the description of the package payload content
  - This way we could preserve current practices and re-use existing software packages
- Command line interface

Conclusions

- Package Manager Service is one of the central middleware services and promptly requested by the users
- Following the feedback from ARDA and input from GAG (and LCG) the AliEn Package Manager was extended to meet the requirements
- First version of gLite Package Manager has been deployed on the prototype and successfully used by ARDA
- Using these two modes of operation combined with lease approach, all use cases can be satisfied
- It has to be integrated with gLite WMS (already works with AliEn Task Queue)
- The version that will fully comply with presented architecture will be a part of the first gLite release