Monitoring of Installed Middleware Packages

Lionel Cons
IT/SDC

15 May 2014

Monitoring of Installed Middleware Packages

- Goal: test that installed middleware packages are at the right version level
- Questions:
  - what is a middleware package?
  - what is a right version?
  - how to test?

Middleware Packages (1/2)

- Example “BDII_site”
- EMI 2 defines “BDII Site v. 1.1.0” as:

```
emi-bdii-site-1.0.0-1
bdii-config-site-1.0.6-1
glite-info-site-0.4.0-1
glite-info-static-0.2.0-1
```

- EMI 3 defines “BDII Site v. 1.2.0” as:

```
emi-bdii-site-1.0.1-1
bdii-config-site-1.0.7-1
glite-yaim-bdii-4.3.13-1
glite-info-provider-ldap-1.4.4-1
glite-info-site-0.4.0-1
glite-info-static-0.2.0-1
```

Middleware Packages (2/2)

- Latest “BDII Site” for EMI 2 is 1.2.1:

```
glite-info-provider-ldap-1.4.5-1
bdii-config-site-1.0.7-1
```

- Latest “BDII Site” for EMI 3 is 1.2.1:

```
glite-info-provider-ldap-1.4.5-1
```

- But later “BDII Top” for EMI 3 contains:

```
glite-info-provider-ldap-1.4.8-1
```

Middleware Packages (MWPs) vs RPMs

- No clear mapping (e.g. EMI 2 ≠ EMI 3)
- Version mapping is even worse
- Some RPMs belong to multiple MWPs
- The manual work to maintain the mappings is time consuming
- Some MWPs may be important to track even if they do not have an entry in BDII
- Idem for system packages like Java

What about using RPMs?

- It is trivial to get the list of installed RPMs
- Collecting installed RPMs can serve multiple purposes:
  - middleware readiness
  - prerequisite testing: OS, Java…
  - security assessment (as done today on WNs)

How to define “right version”?

- A given software version is either:
  - tested and working fine (aka “**GOOD**”)
  - tested and not working fine (aka “**BAD**”)
  - not tested (aka “**UNKNOWN**”)
- Heuristics could be used:
  - if *v1* is “**GOOD**” and *v2* is “**UNKNOWN**”
    and *v3* is “**GOOD**” and *v1* < *v2* < *v3*
    then *v2* could be assumed to be “**GOOD**”
  - if *v1* is “**UNKNOWN**” and *v2* is “**BAD**” and *v1* < *v2*
    then *v1* could be assumed to be “**BAD**”

How to test?

- Volunteer sites report the RPMs they use
- Validation tests are run continuously
- Test results are used to “tag” versions
  - this needs a mapping from tests to RPMs
  - and a policy for conflicting results
- This allows to automatically assess middleware packages versions
  - a human being could bring improved quality

How to use the results?

- All sites report the RPMs they use
- The RPMs are assessed using the acquired knowledge
- Site readiness can be derived from all this

Implementation details

- Pakiti (currently used for WNs security assessment) or similar could be used
- Sites could use a different name/tag to flag their validation machines
- The collected raw data would not be public
- The derived knowledge would be public