ApacheCon Europe
Sinsheim, Germany 5th-8th November 2012

OFBiz Addons goals, howto use, howto manage

Nicolas Malin,
Nov. 2012

Agenda

- History of a birth
- Addons principle
- Addons and their environment (extensive program)
- Conclusion

Once upon a time …

- The history beginning in 2005, with Neogia project
  - To create a B2B ERP for middle size entreprise and follow OFBiz evolution
  - Problem : How to produce the needed improvements (technical and functional) and be involved in OFBiz community (not same business process vision)

Act 1 : patches

- Each improvements have been set on a patch file (generate by svn diff)
  - We have a big patch list with all improvements
  - Apply the list by script on OFBiz
- Result → defeat
  - Difficult to manage patches with consistency
  - Too much time to follow OFBiz evolution
  - Manage clash between two or more patches impossible

Act 2 : Subversion

- We fork OFBiz on other svn deposit and synchronise OFBiz on root branch
- Result → Defeat
  - When loading a new OFBiz version on root branch, many clashes were present so many corrections needed to be done
  - Difficult to separate improvements
    - Difficult to contribute to OFBiz without wast many time

Act 3 : addons

- On 2010 start project Addon manager
  - Create a tool to help resolve problems
  - Switching existing code to an addon structure
  - Planed the end of the Neogia project and the first experimentations
  - Work in progress … the reason of this conference ;)

After 2 years of development

- At this time, addons give the possibility to
  - Follow OFBiz evolution and make it easier to contribute
  - Share improvements for each customer project with separation of specific code
  - Simplify consultant work by dedicating addon by task, help administrator system to deploy customer OFBiz version.

Sinsheim, Germany 5th-8th November 2012

Addons principle : purposes (1/2)

- Follow all modifications realized, since “OFBiz snapshot”
  - Manage addons stacking
- Use similary commands to svn
  - status, diff, revert, ...
- Give a « binairy » format with a version number
  - name-addon-xx.yy.zz.zip

Addons principle : purposes (2/2)

- Include text and binary files on adding, modification et deleting
- File format have to permit to be check by human
- Resist to a clash of modifications
  - Fusion of multiple patch from same file
- Manage all addon dependences

Working

- We work with a tool named **Addon manager**
- OFBiz environment initialization
  - Create a register base : OFBiz snapshot
- Any modification is identified, we can on it:
  - Record an addon
  - Generate a patch
  - Cancel modification

An addon ?

- A modification technical or functional
  - A corrected label
  - A business component
  - A server configuration
- A series of patches UNDER VERSION that progress in the time
- A series of patches :
  - Are in the same directory than the modified file (addon tree);
  - Has an index that give the order of application

Sinsheim, Germany 5th-8th November 2012

Details on the concept of patch

- Two methods of creation depending on files for modifications
  - By diff command (pop)
    - Used by default for all text file
    - The result file will keep diff result
  - By semantic (dop)
    - Used for XML files to update DOM structure.
      - Two instructions : Add and Delete
- Addon may content another type of patch file

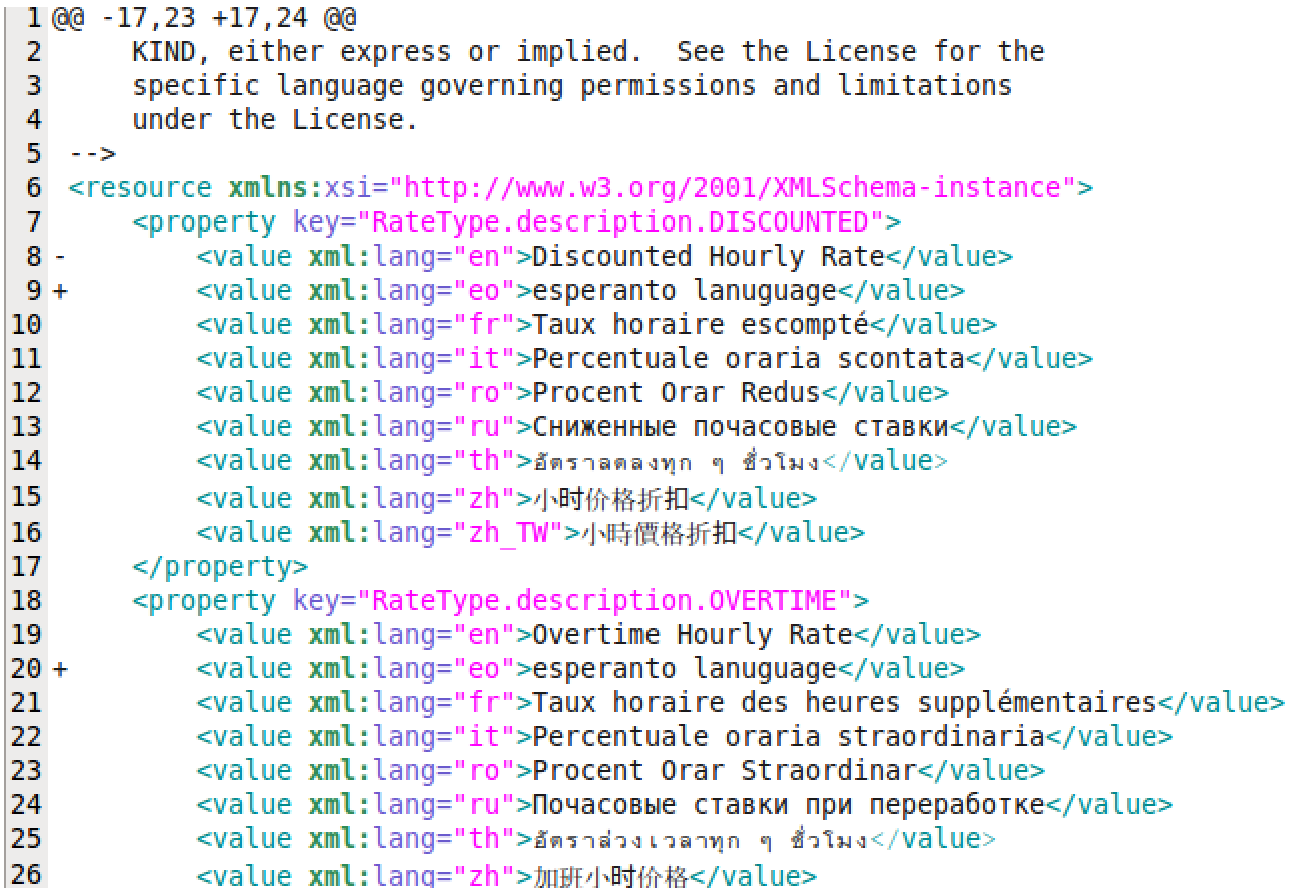
Other file types present on addon tree

- Adding a file (cfp) :
  - Contains the entire file to create
  - Can be managed by hand
- Delete a file (dfp) :
  - The file is empty (used only to give the path file to delete)
  - Can be managed by hand
- Adding a binary file (bip) : similar to CFP for binary
  - Picture, jar, tar etc…
  - Content is unreadable by human
  - Can't be managed by hand

Example

- On addon order-management we can find :
  - ./applications/order/widget/ordermgr/OrderSimpleForms.xml.0.cfp.patch
  - ./applications/order/webapp/ordermgr/WEB-INF/actions/order/OrderView.groovy.0.pop.patch
  - ./applications/order/servicedef/services.xml.0.dop.patch
- We could find that too (but not in this case)
  - ./applications/order/webapp/ordermgr/order/ordershippinginfo.ftl.0.dfp.patch

Pop Example



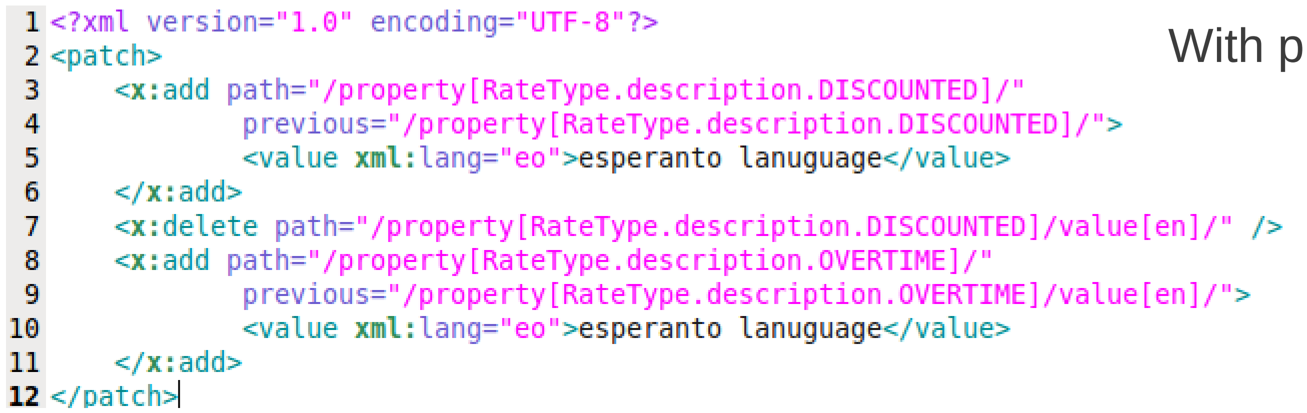
```
@@ -17,23 +17,24 @@
      KIND, either express or implied.  See the License for the
      specific language governing permissions and limitations
      under the License.
  -->
  <resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <property key="RateType.description.DISCOUNTED">
-         <value xml:lang="en">Discounted Hourly Rate</value>
+         <value xml:lang="eo">esperanto lanuguage</value>
          <value xml:lang="fr">Taux horaire escompté</value>
          <value xml:lang="it">Percentuale oraria scontata</value>
          <value xml:lang="ro">Procent Orar Redus</value>
          <value xml:lang="ru">Сниженные почасовые ставки</value>
          <value xml:lang="th">อัตราลดลงทุก ๆ ชั่วโมง</value>
          <value xml:lang="zh">小时价格折扣</value>
          <value xml:lang="zh_TW">小時價格折扣</value>
      </property>
      <property key="RateType.description.OVERTIME">
          <value xml:lang="en">Overtime Hourly Rate</value>
+         <value xml:lang="eo">esperanto lanuguage</value>
          <value xml:lang="fr">Taux horaire des heures supplémentaires</value>
          <value xml:lang="it">Percentuale oraria straordinaria</value>
          <value xml:lang="ro">Procent Orar Straordinar</value>
          <value xml:lang="ru">Почасовые ставки при переработке</value>
          <value xml:lang="th">อัตราล่วงเวลาทุก ๆ ชั่วโมง</value>
          <value xml:lang="zh">加班小时价格</value>
```

Semantic patch explanation

- Don't use diff and patch command
- Works on XML file with a true DOM
- Easier to change by hand
- Very flexible => less time for maintenance

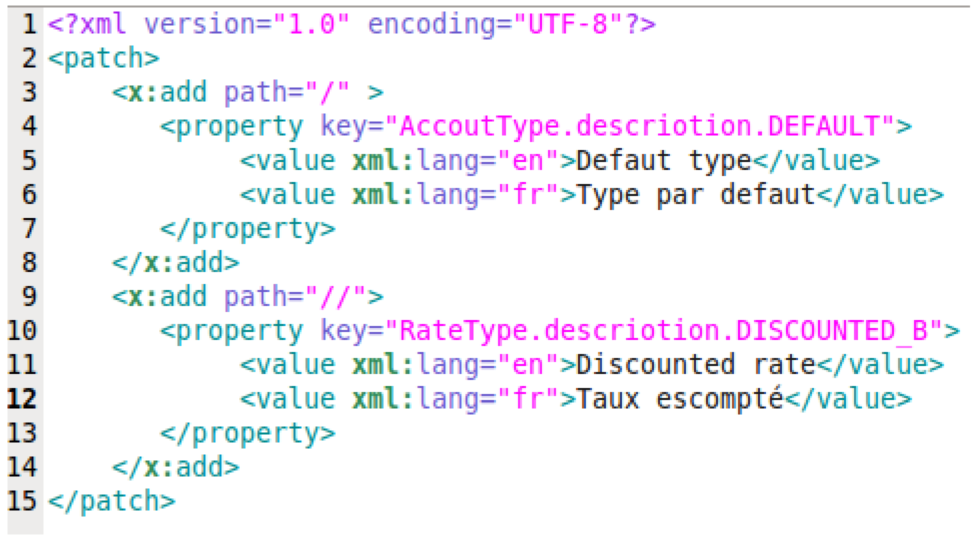
Dop Example

With precise path Dom



```xml
<?xml version="1.0" encoding="UTF-8"?>
<patch>
    <x:add path="/property[RateType.description.DISCOUNTED]/"
            previous="/property[RateType.description.DISCOUNTED]/">
            <value xml:lang="eo">esperanto lanuguage</value>
    </x:add>
    <x:delete path="/property[RateType.description.DISCOUNTED]/value[en]/" />
    <x:add path="/property[RateType.description.OVERTIME]/"
            previous="/property[RateType.description.OVERTIME]/value[en]/">
            <value xml:lang="eo">esperanto lanuguage</value>
    </x:add>
</patch>
```

Without precise path Dom



```xml
<?xml version="1.0" encoding="UTF-8"?>
<patch>
    <x:add path="/" >
       <property key="AccoutType.descriotion.DEFAULT">
            <value xml:lang="en">Defaut type</value>
            <value xml:lang="fr">Type par defaut</value>
       </property>
    </x:add>
    <x:add path="//">
       <property key="RateType.descriotion.DISCOUNTED_B">
            <value xml:lang="en">Discounted rate</value>
            <value xml:lang="fr">Taux escompté</value>
       </property>
    </x:add>
</patch>
```

Addon management

- Use Addon manager
- Adding and deleting on OFBiz environment will be atomic
  - Environment stability
- Possibility to update the addon with a new modification (if open for writing)

Sinsheim, Germany 5th-8th November 2012

Precision on Addon Manager

- Java application on jar format allowing addon manipulation on OFBiz environment
- It runs on OFBiz environment root
- We use acronym ADM

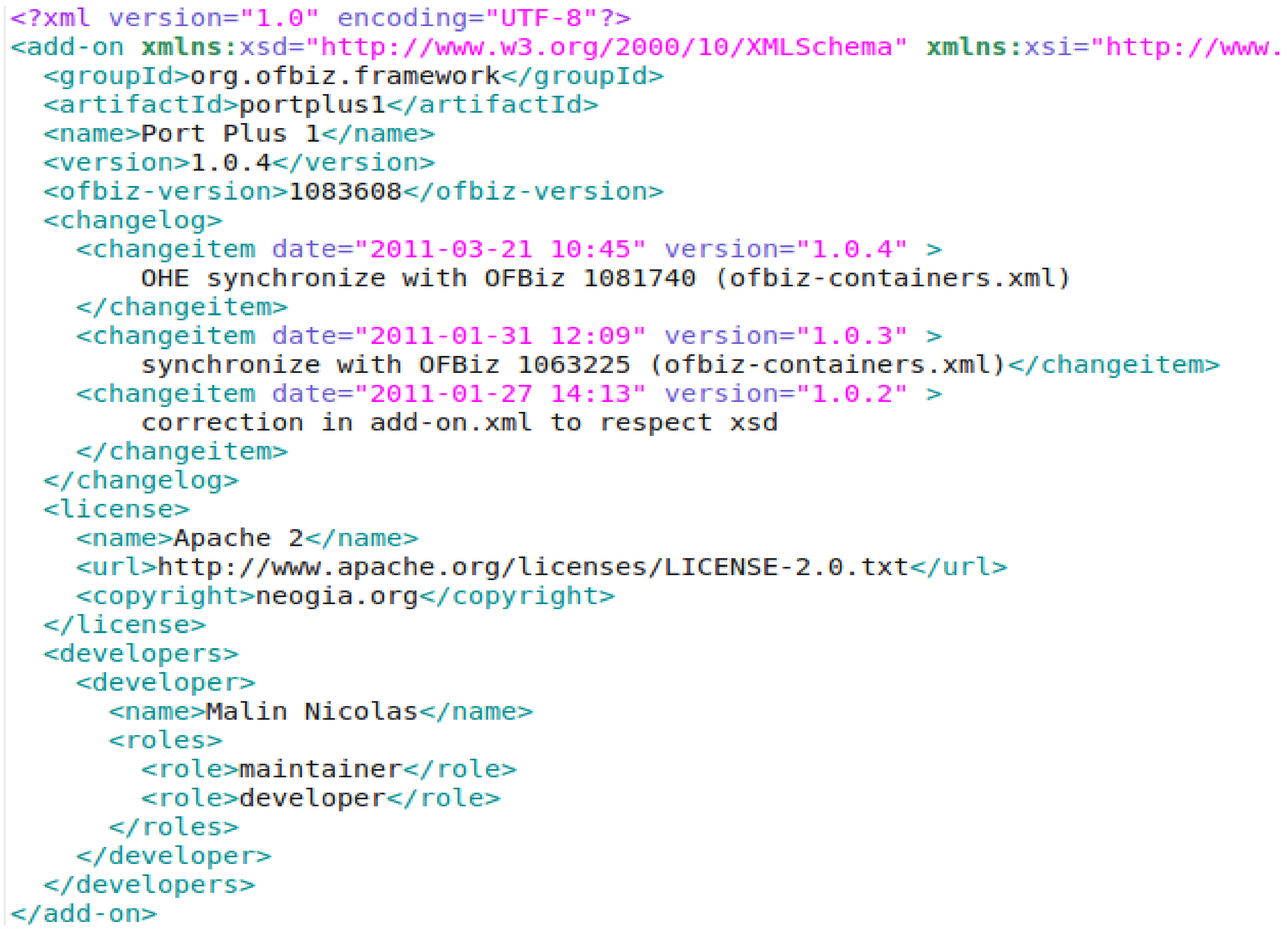
ADM commands

- List addons presents on the environment : **adm list**
- Install an addon from local repository : **adm install chemin**
- Install an addon from a referential : **adm install [-v version] addon**
- List files having in progress modifications : **adm status**
- See modifications performed on a file : **adm diff file**
- List files on an addon : **adm list-file**
- Uninstall addon from the environment : **adm uninstall addon**
- Add a file to an addon : **adm add-file file**
- Delete a file to an addon : **adm remove-file file**
- Restore a file : **adm revert file**
- Update file : **adm update file**
- Package and publish an addon : **adm seal version**
- List addons changing a file : **adm which-addon file**
- … and more : **adm help**

Addon description

- Each addon have information about its content in :
  - a file add-on.xml
    - Addon name, version, minimum ofbiz version, maintainer, license et change log
  - A directory helpdata
    - Content information on how to do to run addon
    - Why it exists
    - Informations permitting to use it (configuration, process, TNR, ...)

Example add-on.xml



```xml
<?xml version="1.0" encoding="UTF-8"?>
<add-on xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" xmlns:xsi="http://www.
  <groupId>org.ofbiz.framework</groupId>
  <artifactId>portplus1</artifactId>
  <name>Port Plus 1</name>
  <version>1.0.4</version>
  <ofbiz-version>1083608</ofbiz-version>
  <changelog>
    <changeitem date="2011-03-21 10:45" version="1.0.4" >
        OHE synchronize with OFBiz 1081740 (ofbiz-containers.xml)
    </changeitem>
    <changeitem date="2011-01-31 12:09" version="1.0.3" >
        synchronize with OFBiz 1063225 (ofbiz-containers.xml)</changeitem>
    <changeitem date="2011-01-27 14:13" version="1.0.2" >
        correction in add-on.xml to respect xsd
    </changeitem>
  </changelog>
  <license>
    <name>Apache 2</name>
    <url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>
    <copyright>neogia.org</copyright>
  </license>
  <developers>
    <developer>
      <name>Malin Nicolas</name>
      <roles>
        <role>maintainer</role>
        <role>developer</role>
      </roles>
    </developer>
  </developers>
</add-on>
```

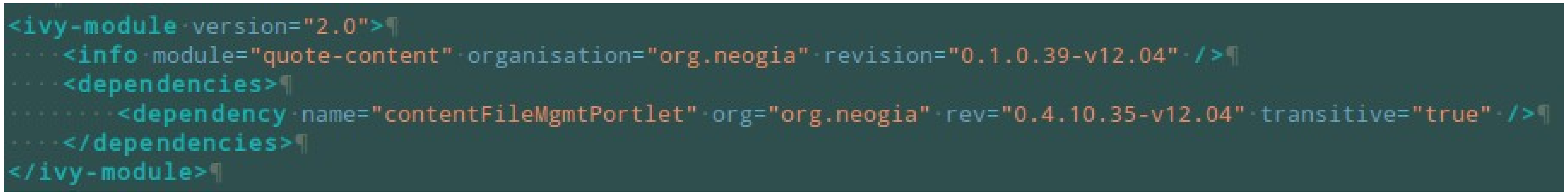
NÉRÉIDE

Addons and dependences

- An addon must contain a limited code
  - Necessity to divide each modification
  - Improve the sharing
    → Causes dependences between addon
- Apache Ivy is used to define and resolve them

http://ant.apache.org/ivy

Example ivy.xml



```xml
<ivy-module version="2.0">
    <info module="quote-content" organisation="org.neogia" revision="0.1.0.39-v12.04" />
    <dependencies>
        <dependency name="contentFileMgmtPortlet" org="org.neogia" rev="0.4.10.35-v12.04" transitive="true" />
    </dependencies>
</ivy-module>
```

Addon and version

- Define by w.x.y.z-v*branch*
  - Ex : 0.1.0.39-v12.04
- The version change :
  - On each marketing announcement (w)
  - On each new functionality (x)
  - On each correction (y)
  - On each dependances upgrading version (z) (explanation after)
- Branch : linked with OFBiz branch

JIRA issues, easier

- It's easier to create a JIRA issue from an addon
  - Apply addon on OFBiz (trunk or latest branch)
  - Run **svn diff** and open your Jira issue
- Depending on suggestions, we can improve this addon and create the issue patch
- If the integration is too long, you can use this addon for your customer project
- After the commit, we remove addon from other addon's dependences

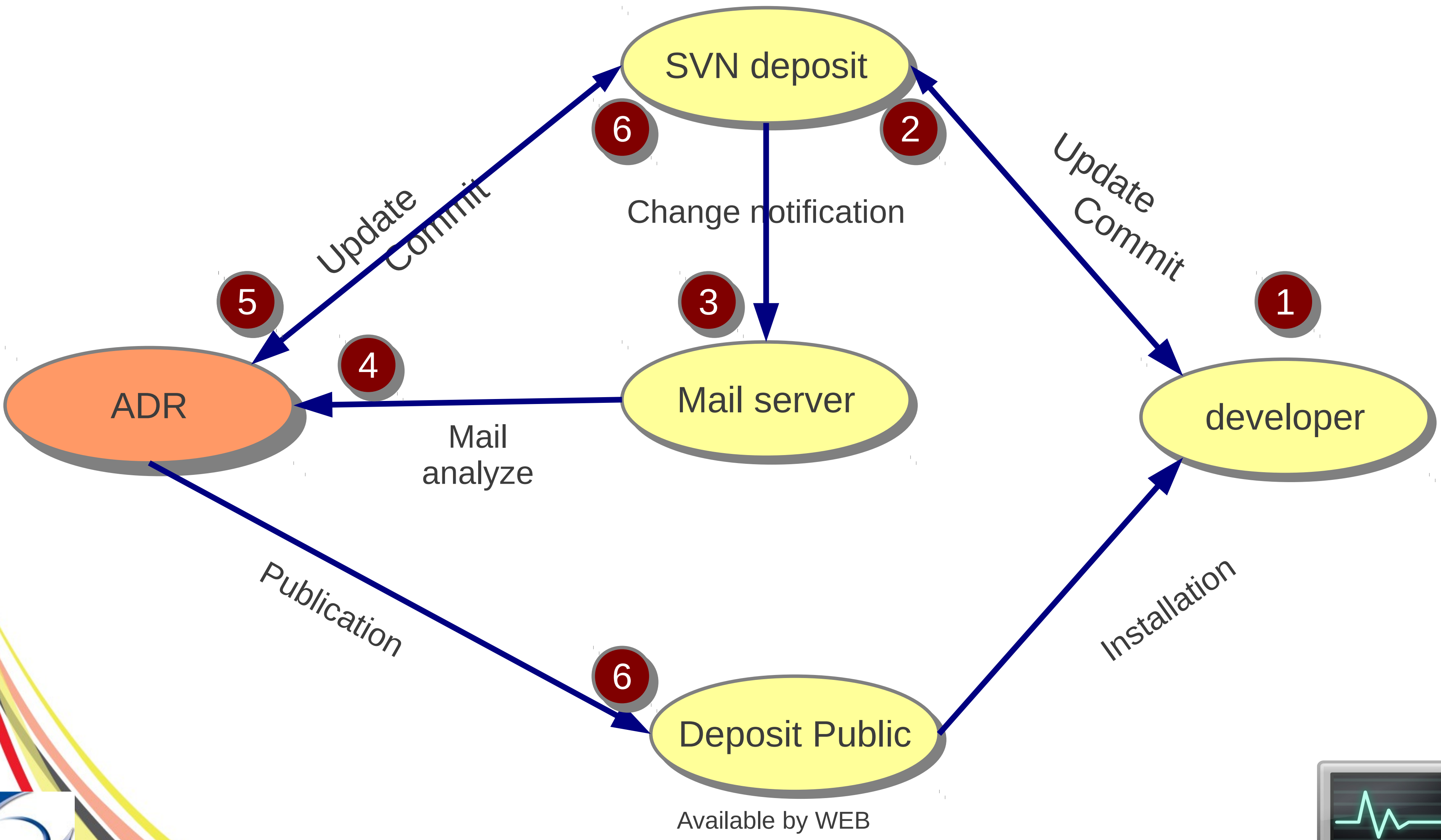
Version and dependence

- New problem : on each addon version change, it's necessary to make follow the version of addon that depends
  - To keep a strict versioned tree
  - To notify all changes
- Impossible to manage it by hand
- Move the problem to an addon repository

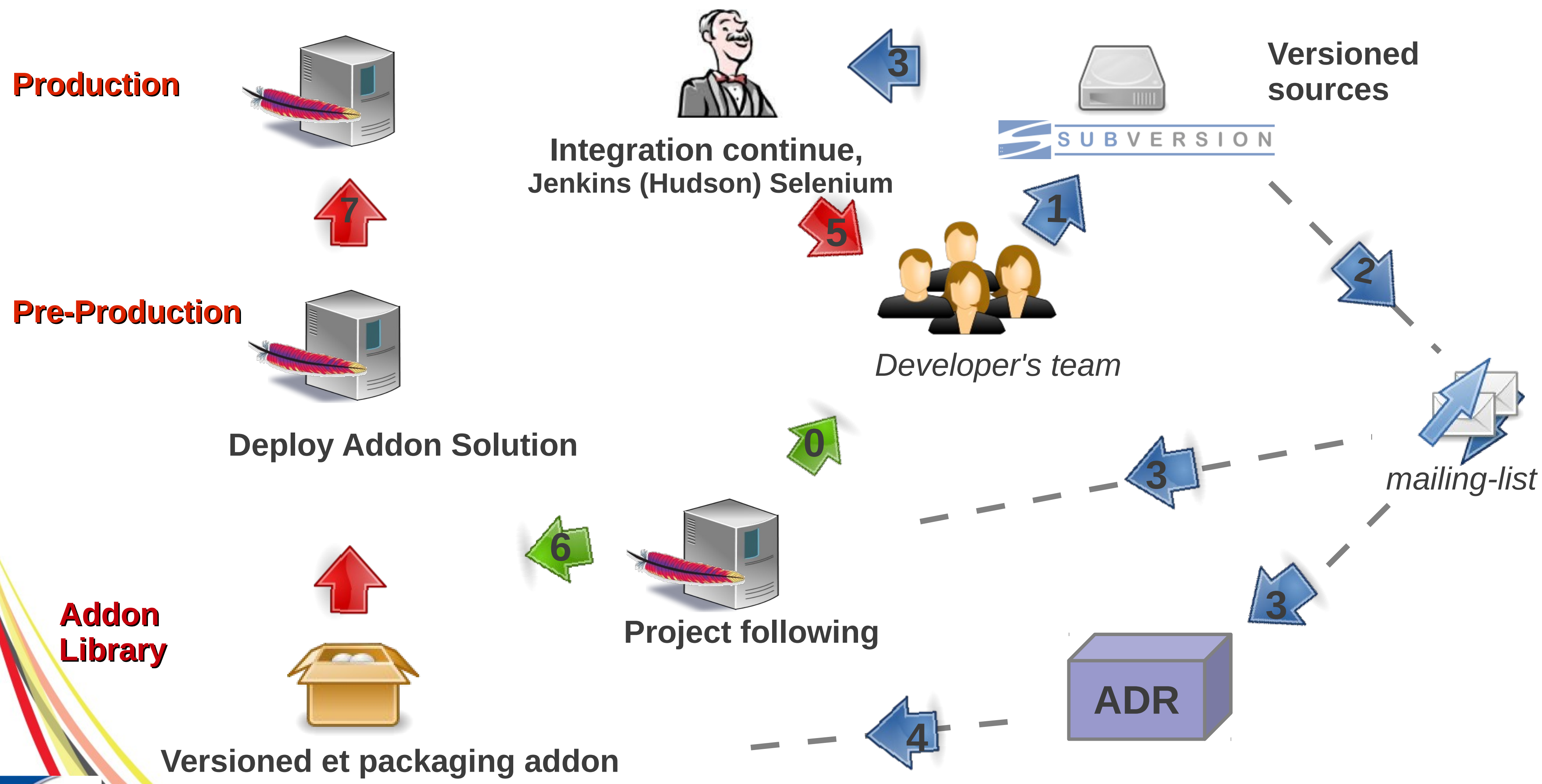
Addon repository

- Why ADR (Addon Repository Manager) ?
  - Each modification implicate a version update
  - Need to update all dependences by recursive process
  - Automatic publication of each new version for ADM resolution

ADR working


SVN deposit
6
2
Update
Commit
Update
Commit
Change notification
5
3
1
4
ADR
Mail server
developer
Mail
analyze
Publication
Installation
6
Deposit Public
Available by WEB

Addon, Adr and Jenkins !



To finish

- Addons resolve our historic issues
  - Working on our own customer's project continuing to follow OFBiz evolution
    - Easier modification application
    - Good segmentation of each improvement
    - Simplify the jira issue creation, help to involve on OFBiz community
- Good identification on which addon version need to be used for OFBiz version

To finish

- However it implicates a rigor and a working method to get a quality result
- We still find methods to work on our own projects management
- There is still so much to discover…

Thanks, and I believe that it's time to eat !