Workflow for development, release and versioning with OSGi / bndtools

Real-world challenges

OSGi Community Event 2012

Dieter Bogdoll, Marian Grigoras
Siemens AG, Corporate Technology
Who we are

Research group within Siemens Corporate Technology (~10 people)

with diverse skills
- computer science
- mathematics
- physics
- mechanical engineering
- electrical engineering

and diverse commitments with respect to development time
- full time
- part time
- work students
- graduate students
- phd students
What we do

Customer solutions (=Siemens business units) based on neural network technologies (and other machine learning algorithms).

Paperware doesn‘t cut it for our customers
→ they want to have software prototypes or solutions

Rapid prototyping (solution) and in parallel build up our frameworks and algorithm toolbox

→ API backward compatibility not that relevant
→ but need reproducibility of old projects
- Java with Eclipse
- Several SVN repositories
- Continuous Integration (Jenkins) to build, test and create deployable artifacts
Artifact for solution deployment

- common for all solutions: launcher
- solution specific configuration
- complete dependency tree of the solution → automate!

Non-devs should be able to create a solution artifact with the latest code

Creation of a solution artifact
- should happen on CI
- should be archived by CI
Requirements for a development workflow

- Enforced modularity
- Use CI to build, test and create artifacts (libs and solution artifacts)
- Partial checkouts (import in eclipse only the project(s) we want to work on, dependencies should be automatically handled)
- Automatic build in eclipse… (easy to use for new team members)
- … and from the command line (mandatory for CI integration)
- Overview of module dependencies, e.g. what classes cause an import (using byte code inspection)
- Tool support for semantic versioning: correct semantic versioning needs manual validation, but tools should make a suggestion
OSGi / bnd / bndtools promise to fulfill all requirements

Goal: leverage OSGi services - start with modularity layer

**bnd**
- generates the *manifest.mf* according to a config file
- no direct manual maintenance of the manifest.mf

**bndtools**
- Eclipse plugin on top of bnd
- Ant tasks for command line / CI integration
The version keyword

Meaning of `version=latest` in the bnd configuration file:
use dependency from workspace if available, otherwise from defined OBRs

Example: ProjA depends on ProjB and ProjC

![Diagram showing dependency between ProjA, ProjB, and ProjC in workspace and OBRs with and without version=latest]

Dev expects to debug the workspace version without the need to deploy to an OBR
Developer OBR (DevOBR)

- OBR holding all bundles developed by the team, as they have been released for team use
- contains only stable versions, no work-in-progress
- may include any number of older versions

DevOBR location:
- Local: in the workspace, checked out from SVN repository
- Remote: accessed via http(s)
- No DevOBR 😊
dependencies must come from source repository

dev working on ProjA must checkout sources of the whole dependency subtree of ProjA

how do you know which dependencies to check out?

dev works with latest (work-in-progress) sources of each lib in the dependency subtree

almost impossible to reproduce the context of a particular build
Local DevOBR

• on new feature/bugfix, dev must release new bundle to local DevOBR and commit the changes to SVN

sadly, all users need to manually update their local OBR copy

deployment: resolving dependencies results in URLs specific to a developer machine => won’t work on another machine
  • dev must create the solution artifact locally
  • non-devs cannot do it

sadly too easy for dev to deploy to customer without committing and versioning the changes
Remote DevOBR

😊 all artifacts in one place

😊 automatic use of newest released bundles

😊 read-only access for devs

😊 preferably only CI has write access into DevOBR
Force source commit to cause a new version bundle?

😊 several commits with the same bundle version

😊 bundle version 1.0.0 in HEAD means work-in-progress for next version

😊 large number of artifacts

😢😢 annoying for devs

Semantic for commit without version bump: let CI check the current status (tests)
Should DevOBR overwrite an existing bundle with the same version?

😊 source repo and DevOBR in sync

😊 default behaviour of bnd

😢😢 over time, multiple artifacts with the same bundle version
Recap: target development workflow

- use `version=latest`
- all checked out sources
- other OBRs

- src repo contains wip
- CI builds / tests / pushes to DevOBR
- remote read-only DevOBR
- does NOT overwrite
- CI can collect deployment artifacts
Typical development sequences

Dev
- wip commit
- no version bump
- A-1.0.0

SVN
- A-1.0.0

CI
- build, test
- release

DevOBR
- A-1.0.0

Do NOT overwrite existing version

release

Release-for-team commit
- bump version
- A-1.0.1

CI
- build, test
- release

DevOBR
- Put A-1.0.1 in DevOBR
Conclusions

• we like OSGi modularity

• using bnd and bndtools feels like the right way to develop OSGi

• for our target workflow we still need tool support for a few features

• surprised there is no default workflow encompassing: dev, source repo, CI, OBR

• would help new teams have an easier start, if the tool suggested (and supported) a default workflow → process started at the bnd hackatron 2012
Thank you for your attention!

Dieter Bogdoll  
email: dieter.bogdoll@siemens.com

Marian Grigoras  
email: marian.grigoras@siemens.com

Siemens AG  
Corporate Technology