Changes

- Release cycle of 6 weeks
- Each release does not have go productive
- Backward bug fixes possible in parallel with new development
- SVN Code re-organization
6 Week release Cycle

6 Weeks release cycle for AWOB Release
• Development of Release X.2 goes in parallel with testing of Release X.1
• Bugs from Release X.1 will be solved during development of X.2 and backported to X.1

What happens in one 6 week cycle:
• Development of new features/bugs/improvements in X.2
• Testing of X.1 and fixing of critical bugs in X.1
• Specification for X.3 (updates/clarification for X.2)

Deployment to production
• Completely independent i.e. release X.x is approved as acceptable for production
• Deployment to production can happen at any time for approved X.x release

Definitions
• Release candidate – software version implementing the agreed scope (new features, bugfixes, improvements, tasks). Release candidate is input to the Test phase.
• Dev – development of new features, improvements, bug fixes, write-up of testing scenarios for new features by developers, developer tests, revision of specification
• Test – testing of a release candidate, revision of testing scenarios
• Spec – specification of new features, initial write-up of testing scenarios
• Release – software version optionally implementing fixes for subset of bugs discovered during Test phase. Is produced at the end of the Test and BugFix phase.

Note: in case of AWOB 0.7 the Week 0-6 is actually Week 0-4 according to the planning for the 0.7 release. All other releases we would plan in 6 weeks timeline.
6 Week Release Cycle (timeline view)

Week zero
- Dev 0.6
- Spec 0.7

Week 1
- Test&BugFix 0.6
- Dev 0.7

Week 4
- Spec 0.8
- Dev 0.8
- Test&BugFix 0.8

Week 10
- Spec 0.9
- Dev 0.9

Week 16
- Test&BugFix 0.9

Releases:
- Release 0.6
- Release 0.7
- Release 0.8
- Release Candidate 0.6
- Release Candidate 0.7
- Release Candidate 0.8
How it works with SVN

1. Development of new version e.g. X.Y is always running from/and code changes are regularly committed to TRUNK

2. When development is finalized:
   a. create a SVN TAG "X.Y-SNAPSHOT" on the trunk
   b. Create a SVN BRANCH "X.Y-SNAPSHOT"

3. Test X.Y version and for each BugFix change:
   a. Commit to SVN BRANCH "X.Y-SNAPSHOT"
   b. Merge the change to TRUNK
   c. Deploy to from BRANCH to QA server

4. When Test & BugFix is finalized
   a. Create a SVN TAG "X.Y" on the BRANCH "X.Y-SNAPSHOT"
   b. Merge unmerged changes to TRUNK

Note: some parts of the process described below may be changed when automated builds will be enabled.
How it works with SVN – Maintenance release

SVN organization

- 3rd-party/
- branches/
- documentation/
- tags/
- trunk/

- ↩
- 0.6-SNAPSHOT/
- liferay-plugins-sdk-6.1.0/

BRANCHES
All BugFix goes here

- ↩
- images/
- manuals/
- scripts/
- sdk/
- settings/

TAGS

- ↩
- 0.6-SNAPSHOT/
- 0.6 (after Step 4 on the right)

Snapshot of trunk: Release candidate
Snapshot of Branches: Release

- State:
  - version in development is e.g. 0.8, release e.g. 0.6 is deployed on prod
- Cause:
  - A bug which must be fixed on production
- Prerequisite
  - Set-up the same environment on the maintenance server as for production server
- Steps:
  - Make a branch from the release deployed on the production (i.e. if deploy was 0.6, make branch 0.6.1-SNAPSHOT)
  - Fix bug and test
  - When done
    - make a tag 0.6.1
    - Merge the change to trunk (of 0.8)
    - deploy on production
What we deploy where

- **awob-dev.mpdl.mpg.de**: (Automated) build from new development
  Source: TRUNK

- **awob-qa.mpdl.mpg.de**: (Automated) build from release candidate and bugfixes
  Source: BRANCHES -> X.Y-SNAPSHOT

- **awob.mpg.de (PRODUCTION)**: Deploy of an approved X.Y Release version
  Source: TAGS -> X.Y

- **awob-test.mpdl.mpg.de (DEMO)**: Deploy of an approved X.Y Release version
  Source: TAGS -> X.Y

- **awob-maintenance.mpdl.mpg.de**: (Automated) build from release deployed on productive and bugfixes
  Source: BRANCHES -> X.Y.Z-SNAPSHOT
  Where X.Y.Z is e.g. AWOB 6.0.1
Eclipse set-up

Two workspaces
- for Trunk
- For Branch
- Make sure commits in Branches are merged to Trunk
- Discipline required from committers
Ending remarks on the Process

- There can be a certain “buffer” period between 2 release cycles
- Not each release has to be deployed on Production i.e.
  - Testing and Bug Fixing finalized -> does not mean approved for deploy to Production
- 6 Week Release cycle of a release X.Y can be shortened only if all agree and release scope is changed accordingly
- What happens: If productive version X.A has a critical bug discovered, and development is in X.D (i.e. there are 2 versions not deployed to productive in between) – MAINTENANCE RELEASE
  - Bug fix has to be made in BRANCH X.A-SNAPSHOT on the same SVN revision tagged as „X.A“
  - Bug fix has to be merged back to TRUNK and to the last branch in testing phase e.g. BRANCH X.C-SNAPSHOT (for which TAG X.C is not yet issued, as it is in test phase)
  - There can be only one BRANCH in testing/bugfix phase at a time (unless productive version bug fix described above must be done)
AWOB & Liferay
AWOB & Liferay

- Themes: AWOB Astro Office Theme (Look&Feel&Layout Structure)
  - css
  - javascript
  - Velocity templates
  - Icon library

- Portlets: so-portlet, web-form portlet, private messaging portlet,
  - Own business logic (logic, services, JSP, javascript, css)
  - AWOB specific data structures (service builder)

- Hooks: AWOB Hook
  - Extension/customization of Liferay built-in portlets (mostly in JSP „logic“ and layout)
  - AWOB specific model listeners
  - AWOB specific actions (i.e. login)

- Extensions: awob-ext
  - Extension/customization of Liferay built-in core services and models
  - Extension with AWOB-core services and models
AWOB Status and Testing

- Intensive development (many changes in logic, GUI, concepts)
- Testing mostly „manual“
- Not yet unit or integration testing
- Rely on Liferay provided functionality (relatively stable)
- Use case/scenario testing descriptions in JIRA
- Not yet continuous integration and automated deployment (keep it low so far intentionally)
  - A word on deployment: with exception of awob-ext – runtime deployment/redeployment possible

Plans:
- Continuous integration
- Unit tests
- Integration tests
- GUI tests