



# Zero Downtime Migration



# Zero Downtime Migration



## Zero Downtime Migration

Polarions concept for providing SCM-migration services without interfering developer teams, by introducing a special migration process, tracking and enforcing it by polarion



## How it works

Migrations are departed into separate steps:

- usermigration
- defining SVN-Repostructure
- deciding which parts to migrate
- defining hook-scripts for process enforcement
- 2 step migration:
  - test migration
  - live migration
- archival storage of old scms data

Zero Downtime Migration will provide tracking of all steps results and history [except archival storage]



# Usermigration

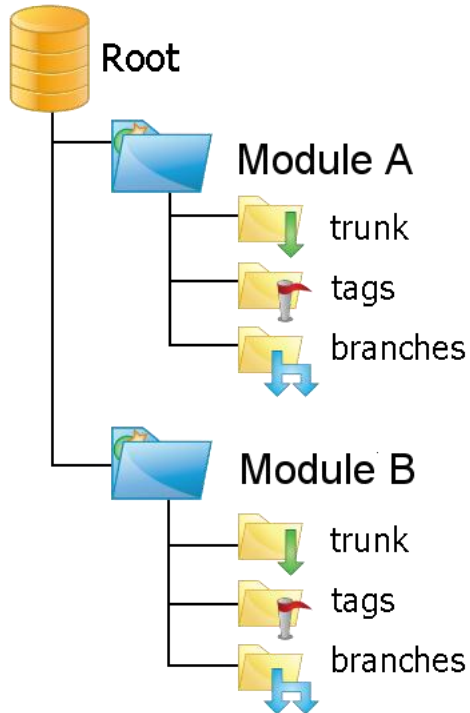
Users and permissions have to be migrated to the new svn-system. There are different possibilities for userauthentication in SVN:



- HTTP-AUTH
- LDAP
- SASL
- SSH

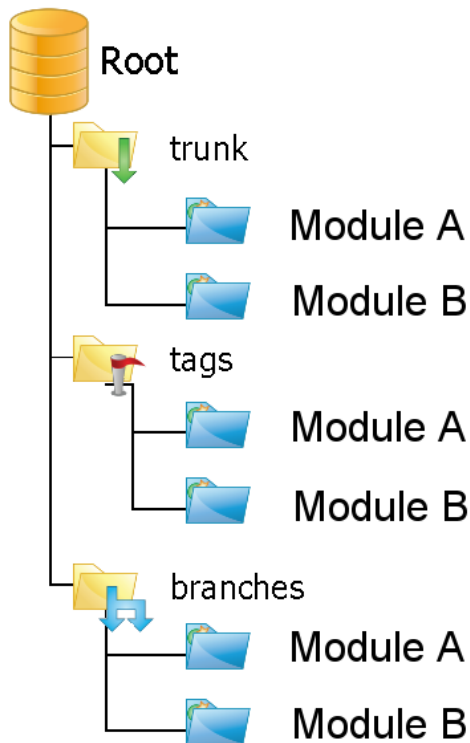


## modules on root



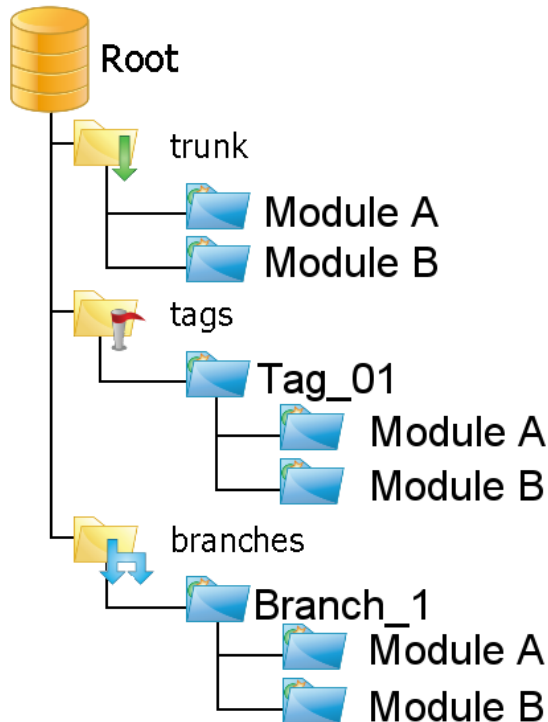
- each module has to be checked out separately
- commits can not span multiple modules
- tags can not span multiple modules in single commit (workaround by 3<sup>rd</sup> party tools)
- „official“ svn layout

**trunk/tags/branches on root; each module will be tagged separately**



- checkout of all modules is possible
- commit can span multiple commits
- tags/branches can not span multiple modules

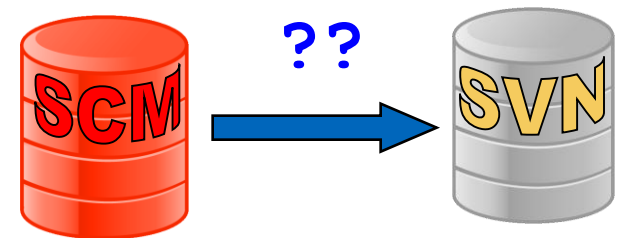
## trunk/tags/branches on root; all modules will be tagged



- checkout of all modules is possible
- commit can span multiple commits
- tags/branches can span multiple modules
- difficult to determine relevant files of a tag

# Migration reasonable ?

- Too large database
- too much insignificant (very old) historic data
- migration tooks too long
- migrated history is unusable
- cannot migrate all project dependencies





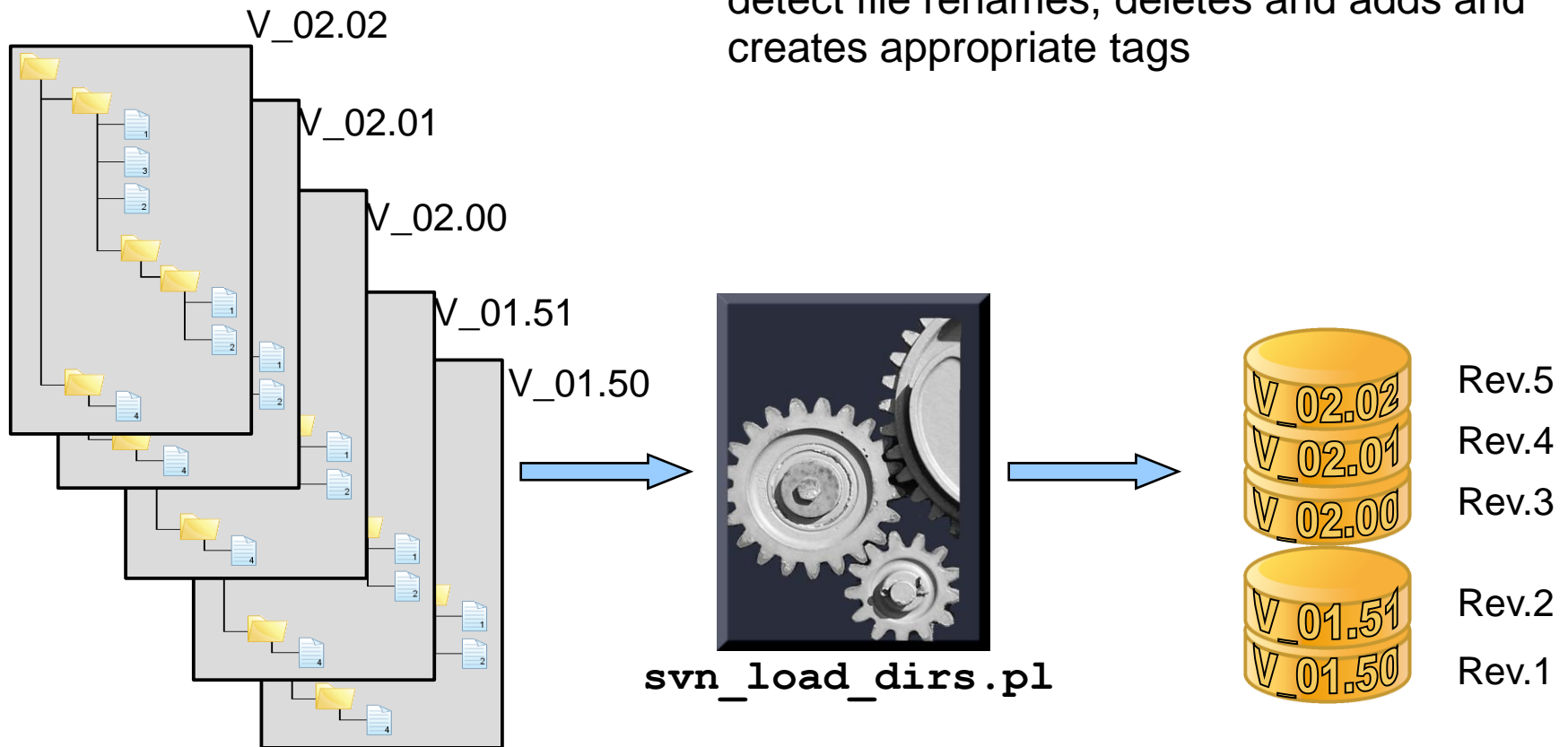
## Alternative 1: migrate only tags ?

Migrate only tags(releases, labels..) and record these as versions:

- + all released versions will be reconstructable in Subversion
- + insignificant(?) history will be removed
- + easy, clean and fast
- + no testing required
  
- no historical data of development between tags
- bugtracking data may be lost
- associations between tasks/bugs and users are lost

# Alternative 1: `svn_load_dirs.pl`

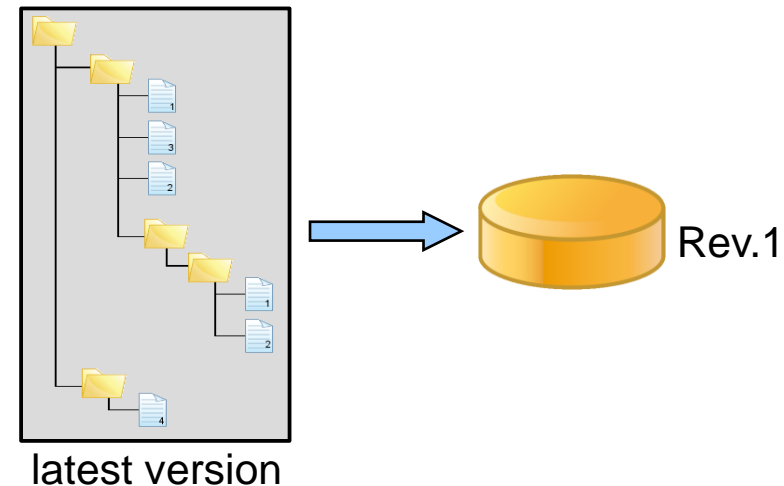
`svn_load_dirs.pl` can be configured to detect file renames, deletes and adds and creates appropriate tags



## Alternative 2: migrate only head ?

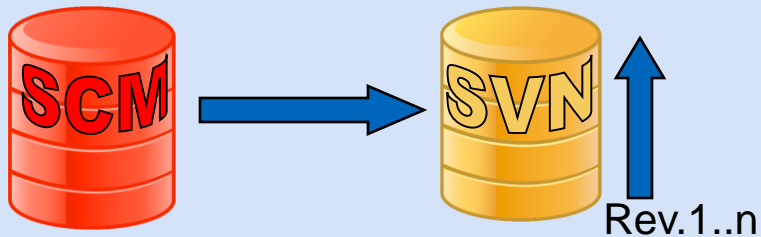
Migrate latest version of project and use old SCM for recreating builds or research

- + No effort at all
- + subversion repository starts clean and small in size
- + subversions new history will be more significant as work continues
- cannot „migrate“ separate (developer/feature)-branches
- license cost for old SCM
- break of toolchain to get old history/releases



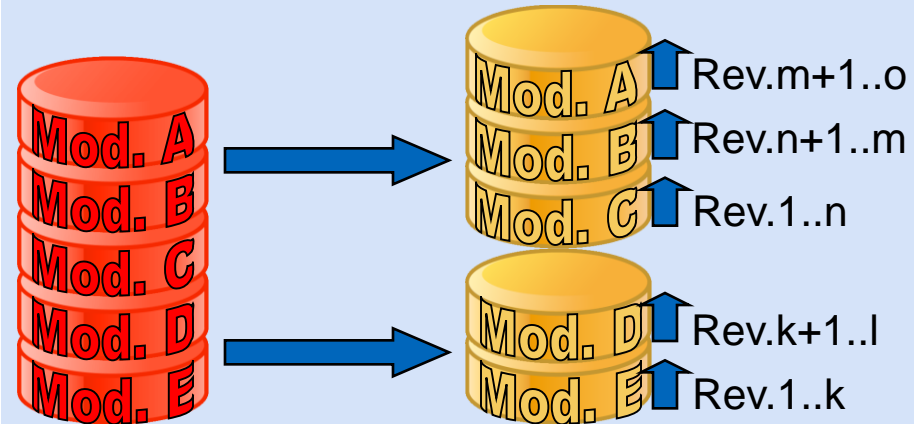
## migrate whole repositories:

- + commits following chronological order
- + mapping of tags/branches is trivial
- you may migrate obsolete data
- no restructuring possible



## migrate per module(project):

- + flexible selection of modules
- + movement of modules into different repositories possible
- commits are „stacked“ per Modul
- commits not in chronological order
- mapping of tags/branches may be not optimal





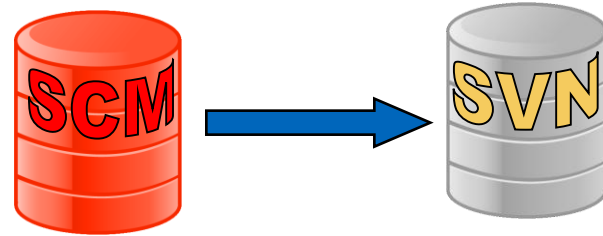
## Hook-script library

Polarion provides an always growing library of hook-scripts to enforce processes widely used in large companies like:

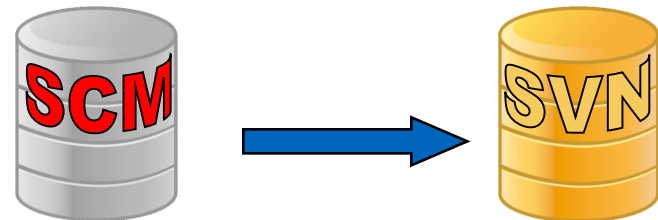
- “Do not commit on tags”-hook
- „No empty commit-message“-hook
- „blacklist commit-message“-hook
- „Notify teams after commit“-hook
- „only integrator can commit on trunk“-hook
- „permission to tag“-hook

## 2-step migration

To ensure continuity for development and migration team, each repository will be **testmigrated** first



- After feedback of development team and the experience with the testrun an appointment for live migration will be set
- So Development team can preview SVN-Repo, testing 3<sup>rd</sup> party tools, builds etc, on real data without stopping to work with your old SCM.
- After **live migration**, development team continues work on subversion



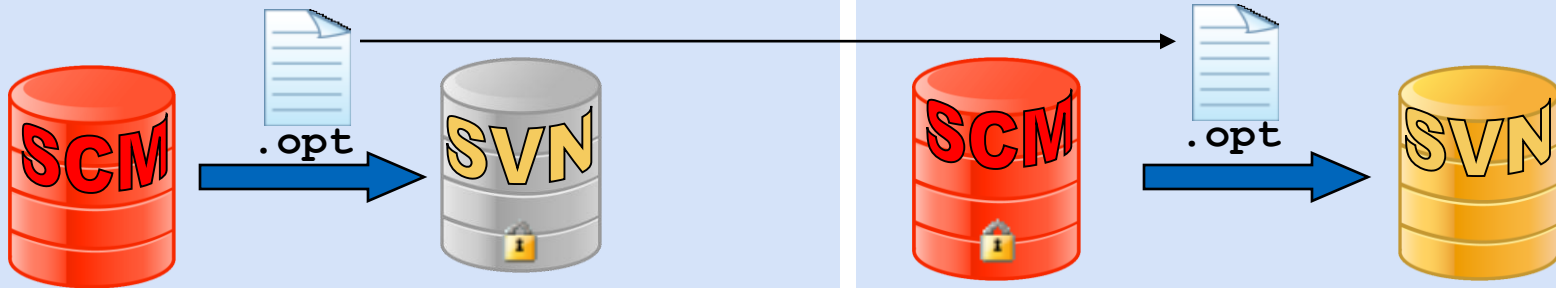
## 2-step migration

### Testmigration:

- After migration svn-repo is write protected for inspection
- development is still working on old scm
- on migration problems, optionfile can be modified (optionfile is stored in polarion tracker)

### Livemigration:

- before migration, write access to scm will be denied
- the optionfile of our last testmigration will be used
- after migration work will continue on new svnrepo
- archival storage of your old scm data, optionfile and migration log



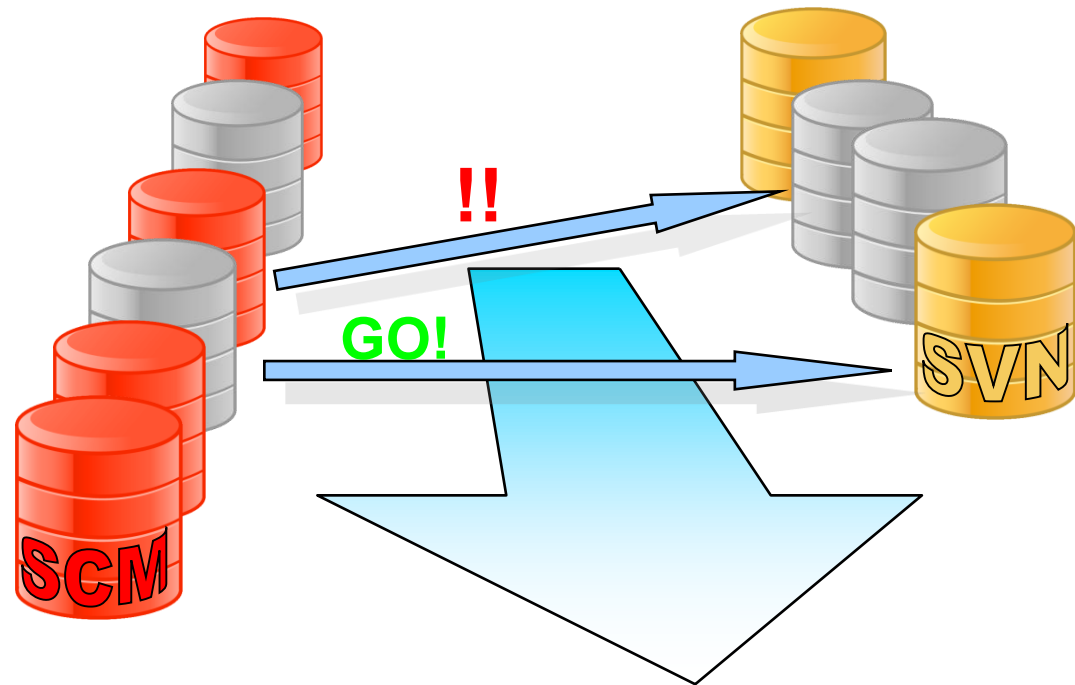
Migration of multiple repositories takes time

different repositories will be  
on different states  
reporting problems

for livemigration all changes  
have to be applied  
according to last test-run

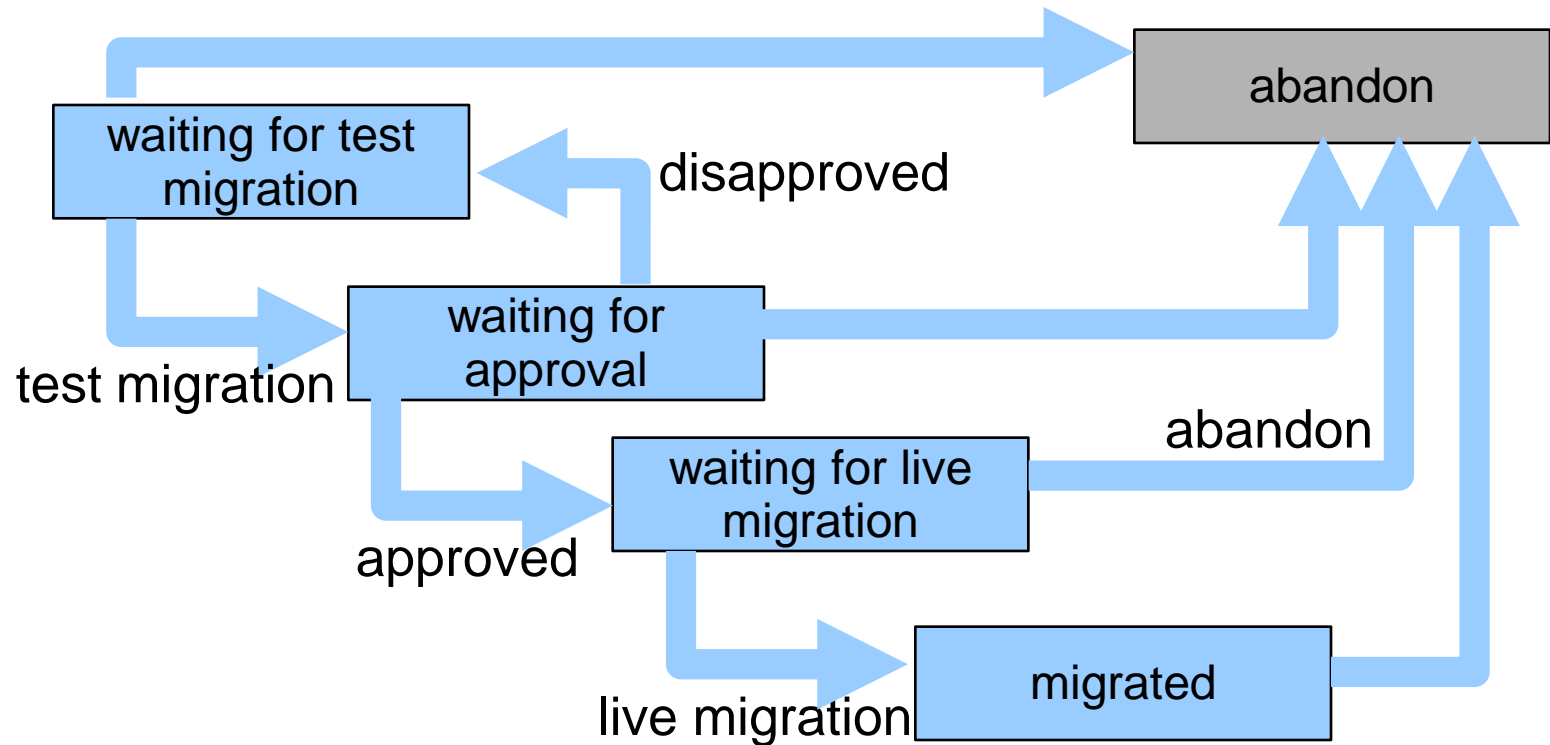
approval can take a lot of time

other repositories have to be  
migrated while waiting for  
approval





# migration workflow





## tracking test-migrations

Polarion will play a significant role on organizing all steps of discussing, tracking and documenting the different steps of the migration process.

Most important is the tracking of the different stages of a repository/module migration in a test- and live stage. Feedback of a testmigration will also be documented into polarion by the development team. If problems are encountered in migration, the optionfile can be modified and another test can be started until the initial problems are gone. All steps and stages are documented

After successful testmigration a livemigration will be started with same optionfile which is stored in polarion repository.



# tracking test-migrations with Polarion ALM

The screenshot displays the Polarion ALM interface. The top navigation bar shows the user as 'System Administrator' with options for 'Logout', 'Help', and 'About'. A search bar is present on the right. The left sidebar contains a 'PROJECTS' section with a filter for 'Selected Projects' and a tree view showing a 'Repository' containing a 'zero\_downtime' project. Below this are 'Shortcuts' and 'Topics' sections.

The main area is titled 'Work Items > zero\_downtime'. It features a toolbar with options like 'Table', 'Live Plan', 'Road Map', 'Multi Edit', 'Matrix', and 'Time Sheet'. A table lists work items with columns for 'Due Date', 'ID', 'Title', 'Status', 'Assignee', and 'Time Point'. The table contains 5 items, with 'ZDM-2' selected. A tooltip for 'ZDM-2' indicates it is 'wait for Live migration - Repository is waiting for live migration'.

Below the table, there are buttons for 'Edit', 'Save', 'Cancel', 'Actions', and 'Auto Suspect'. A 'Normal Form (all fields shown)' is displayed for the selected item, showing details such as 'Type: Repository', 'Severity: Normal', 'Author: System Administrator', 'Project: zero\_downtime', 'Initial Estimate: 3d', 'Assignee: Robert Projec', 'Status: wait for Test migrat', 'Resolution: wait for Test migration', 'Priority: Perform action Test migrate', and 'Due Date: 2008-10-08 00:00'. The 'Time Point' is 'Beta (2008-11-02)'. The form is created on '2008-10-04 05:16' and updated on '2008-10-04 05:36'.

At the bottom, the footer contains copyright information: '© Polarion Software 2004-2008', 'Powered by Polarion® ALM for Subversion™ 3.1.2. EVALUATION LICENSE (88 days left), licensed to: Polari', and 'Feedback/Suggestions | Support'.



**Any questions?**