Migration from CM Synergy to Subversion at UBS

Michael Diers — elego Software Solutions GmbH

CM Synergy vs Subversion

Considerations

Approach

Selected issues

Tools

Recommendations

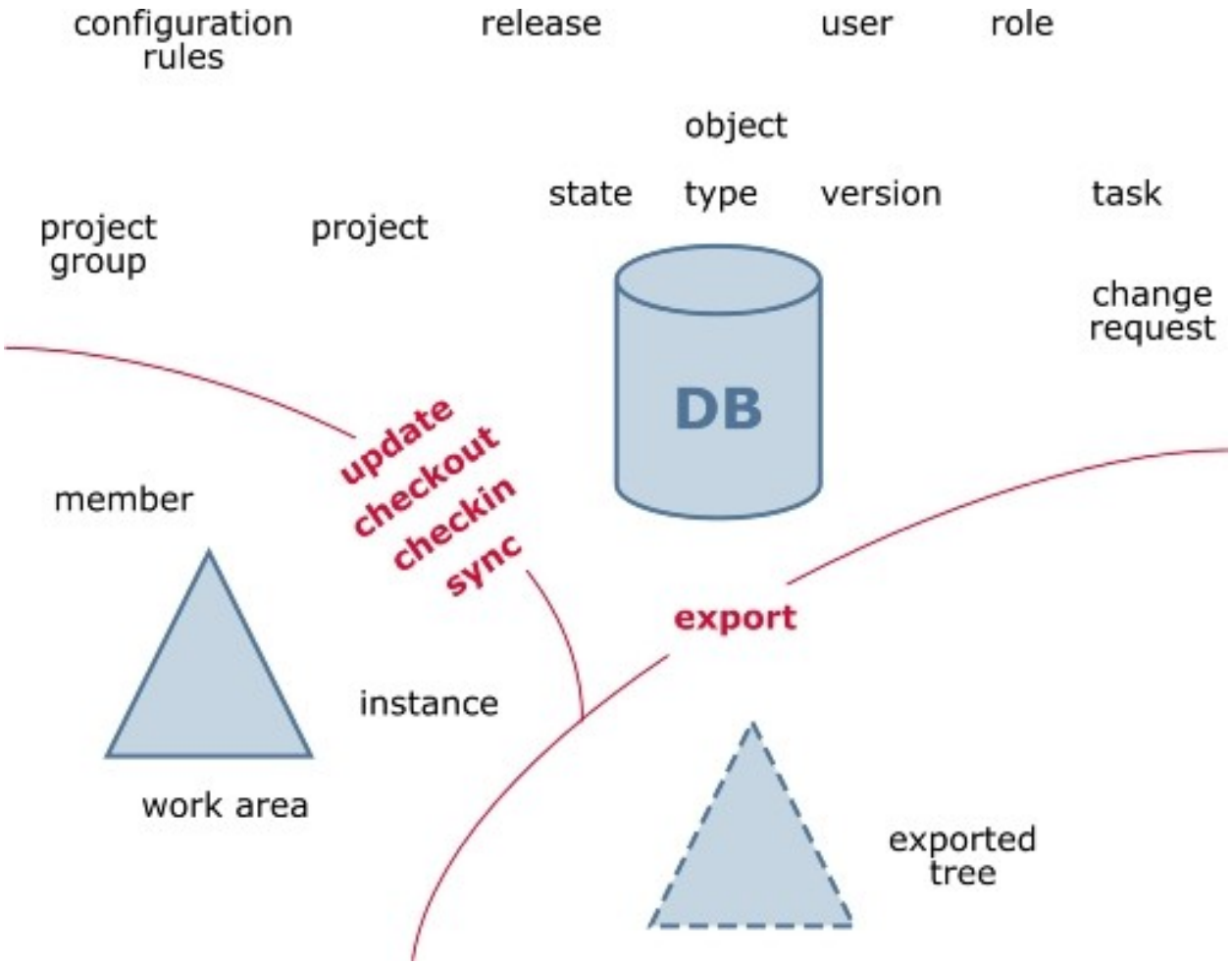
CM Synergy: Concepts

- Change Management System
  - Not just version control
  - Process oriented, with well-defined workflow
  - Up-front preparation steps required for each project
- Projects
- Tasks
- Roles
  - Build Manager, Team Lead, Developer

CCM: CM Synergy CLI (Continuus Change Management)

SVN: Subversion CLI

CM Synergy: Overview



CM Synergy: Data model

- All metadata is stored in a relational database
  - Must query DB to aggregate fragments of data into "useful" information
- A *project* contains directories, files and other projects
- Any set of changes is a *task* within some project
- Every *version* of a file or directory is stored once
- Exact version of file in a directory is context dependent
- Context is provided by a *project*
  - Lack of a uniform global context makes automation difficult

CM Synergy: Key differences from Subversion

- CCM is complex!
- To be fair, it does a lot more than SVN
  - CCM is an integrated change management system
  - SVN separates mechanism and policy: version management toolkit + best practices
- CCM needs project/task context to find versions/changes
  - No global revision numbers
  - No URLs
- Checking out a project in CCM modifies the server state
  - Keeps server-side information about attached workspaces: to be avoided for migration

Considerations

- Teams use CM Synergy in different ways
  - Complex system, requires proper user training
  - Extreme example: one project never made a release; metadata stays in a “transient” state, cannot be reported via CCM query language
- When CCM best practices are not followed
  - Retrieval of history and branching structure gets quite involved
- Extent of data migration
  - Development teams want different levels of detail to be preserved during migration

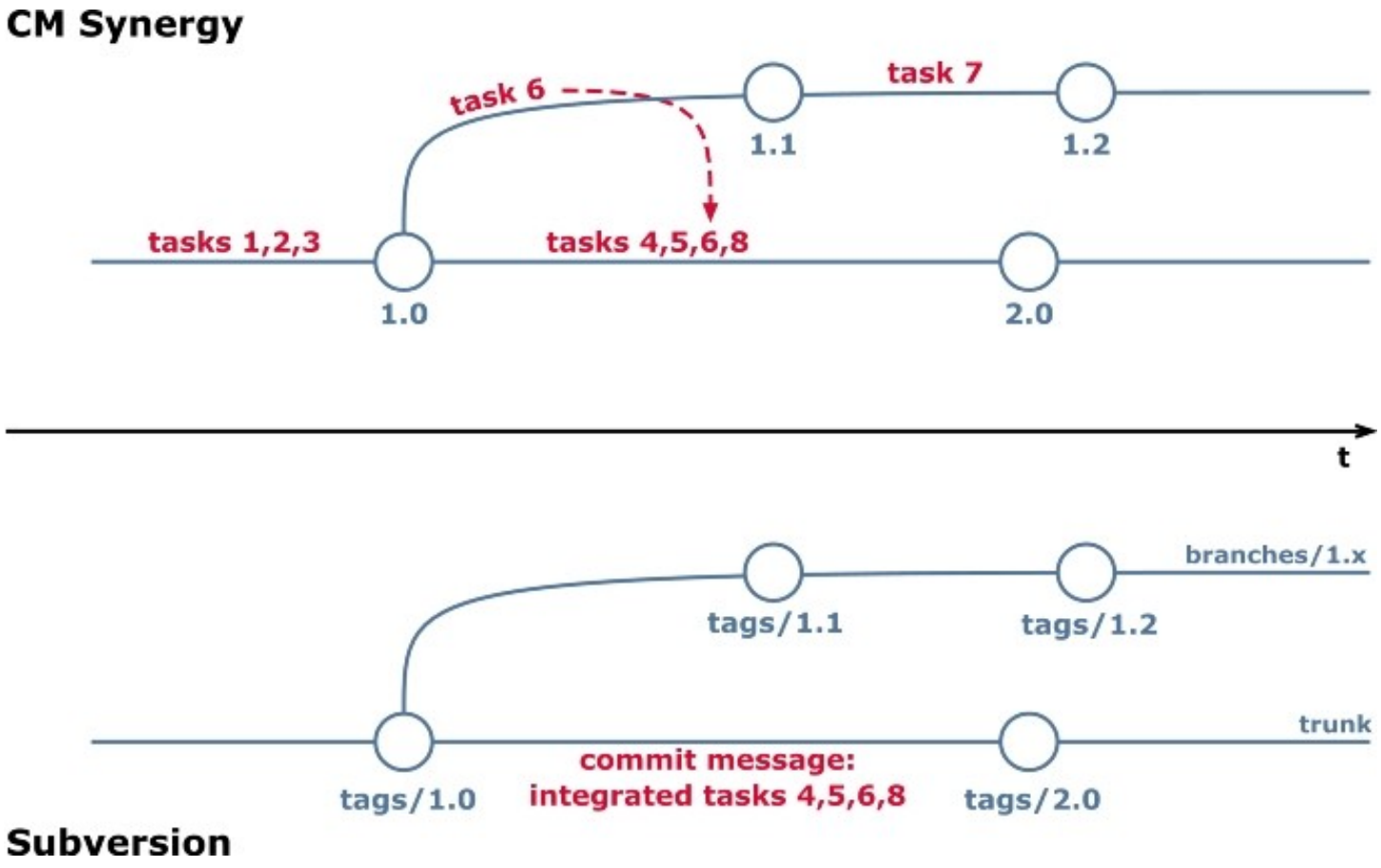
More considerations

- Size of source repositories in CCM
  - 0.1MB to 600MB per release
  - Migration process can take a long time when many releases are involved
- Information retrieval from CCM
  - Path of a particular file revision in a project structure
  - Relationship between projects and subprojects
  - Time line of events
- May be able to directly mine CCM database for metadata
  - Not done here, used CLI instead
  - In retrospect: perhaps not a bad idea

Approach

- Different project needs
  - Cost and effort varies
  - *No silver bullet,* multiple migration procedures
- Release-based migration
  - Preserves information about releases
  - Aggregates changes between releases into one change set
- File-based migration
  - Track changes at file level, ignores structural changes
  - Preserves commit history along the time line
  - No grouping of changes according to associated task
- Combine the two basic approaches

Approach: Baseline Conversion



Issue: Timestamps

- CM Synergy (both GUI and CLI) does not support UTC for timestamps
- The timezone is never printed by command output
- For some commands, the client timezone is used
- For others, the server timezone is used
- If server and client are in different time zones, you have a problem
- Luckily for us, Switzerland is a small country!

Issue: Data extraction

- Some data can be retrieved by an ad-hoc query language
- Some data can be retrieved only by peculiar administrative commands
- Some data cannot be retrieved by any command
- The GUI client has features not exposed to the command line
  - Automatic resolution of project/subproject relationship
  - Displays information about objects in “transient” state

Tools

- GNU tool chain and other Free Software
  - No administrative privileges required
  - Cygwin (Windows)
  - OpenPKG (Solaris 8)
- Perl
  - Portable, extensive library (CPAN)
  - Algorithms for dealing with graph/tree data structures
  - CCM wrapper VCS::CMSynergy
- svn_load_dirs.pl
  - Small enhancement to set author and date of each revision

Recommendations

- Batch CCM operations
- Cache information locally for later migration steps
  - File system (file contents)
  - SQLite database (structure and metadata)
- Run migrations on CCM server
  - Close to physical data, avoid LAN traffic
- Work around “given” situation
  - Windows: turn off antivirus products
  - Avoid having CCM databases still in production, else must have restartable scripts to handle e.g. periodic system downtime for backups

Summary

- CCM and SVN have very different data models; however, you may not have to migrate all of the CCM data to SVN
  - Some metadata should go into issue tracker instead
- CCM is complex and its CLI is somewhat inconsistent; enter Perl and VCS::CMSynergy
- Every team seems to use CCM differently; luckily, SVN has flexible mechanisms and reasonable best practices
- There's nothing wrong with a simplistic approach: doing release-based migration to SVN and keeping CCM around for reference in r/o mode

Thank you!

Collaborators

- Petra Gratzer
- Lars Soltau
- Stephen Butler

Contact Information

- Michael Diers <mdiers@elego.de>

elego Software Solutions GmbH
Gustav-Meyer-Allee 25
13355 Berlin
Germany

http://www.elego.de/