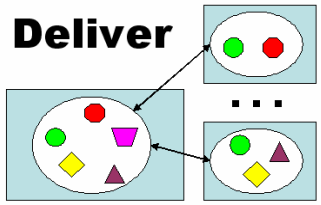
Binary Change Set Composition

Tijs van der Storm
Centrum voor Wiskunde en Infromatica
storm@cwi.nl



Introduction

- Me: Tijs van der Storm
- Phd Student, project *Deliver*:
  - Intelligent Knowledge Management for Software Delivery
- My focus: software configuration management
- This talk:
  - Goal: efficient, lightweight, generic upgrading of component-based applications
  - How: binary change set composition

Perspective: continuous delivery

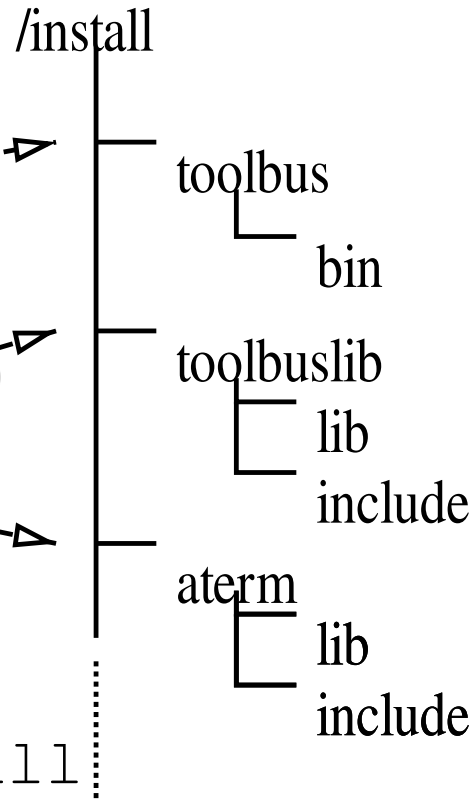
- Continuous...
  - Integration
  - Release
  - Updates (this talk)
- Setting: heterogeneous component-based applications
- Topic: binary change set composition
- Steps towards self-updating software

Step 1: building components

- Build in topological order:

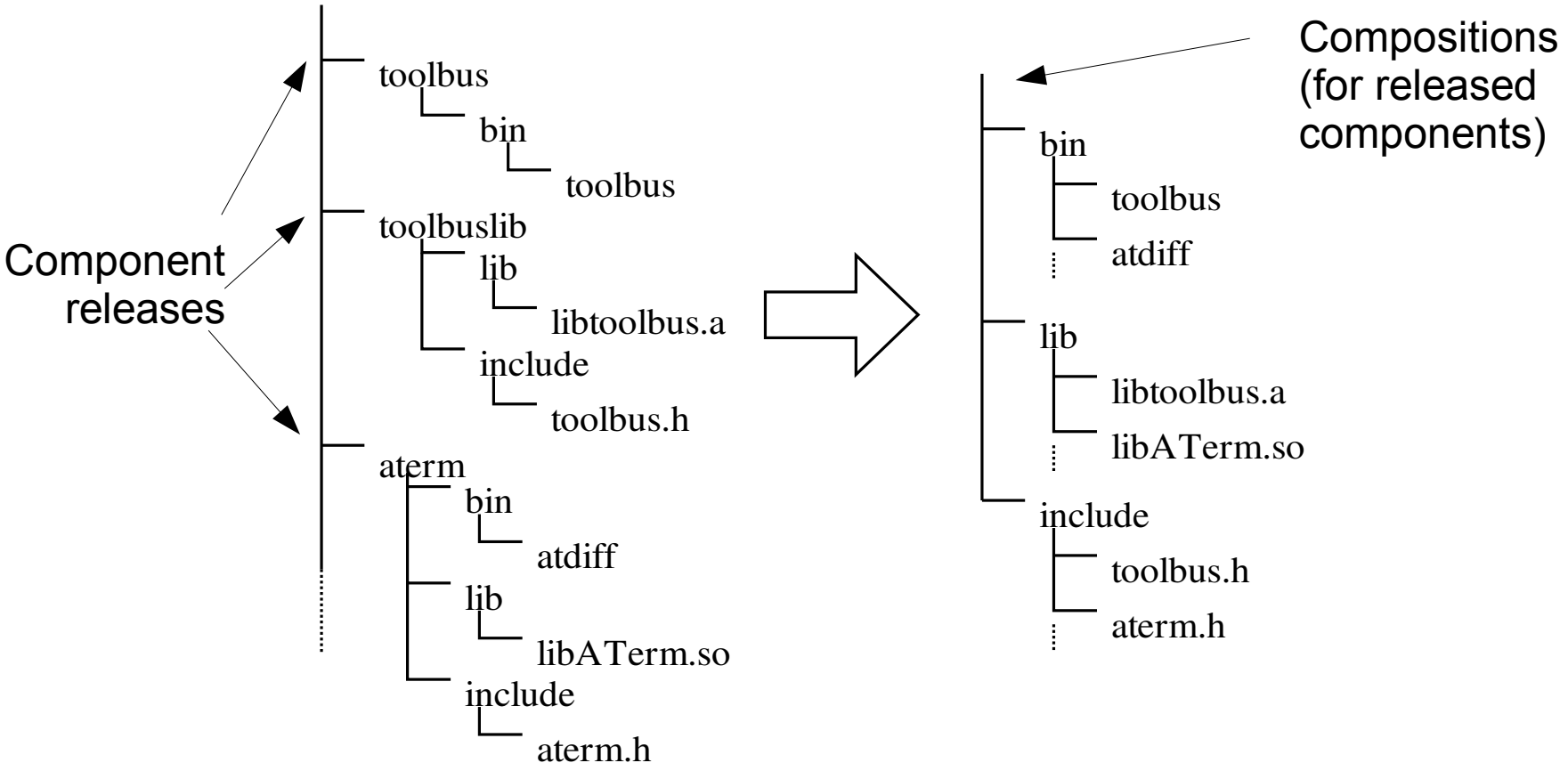
```
./configure
 --prefix=/install/toolbus

 --with-toolbuslib=/install/toolbus-lib
 --with-aterm=/install/aterm
```

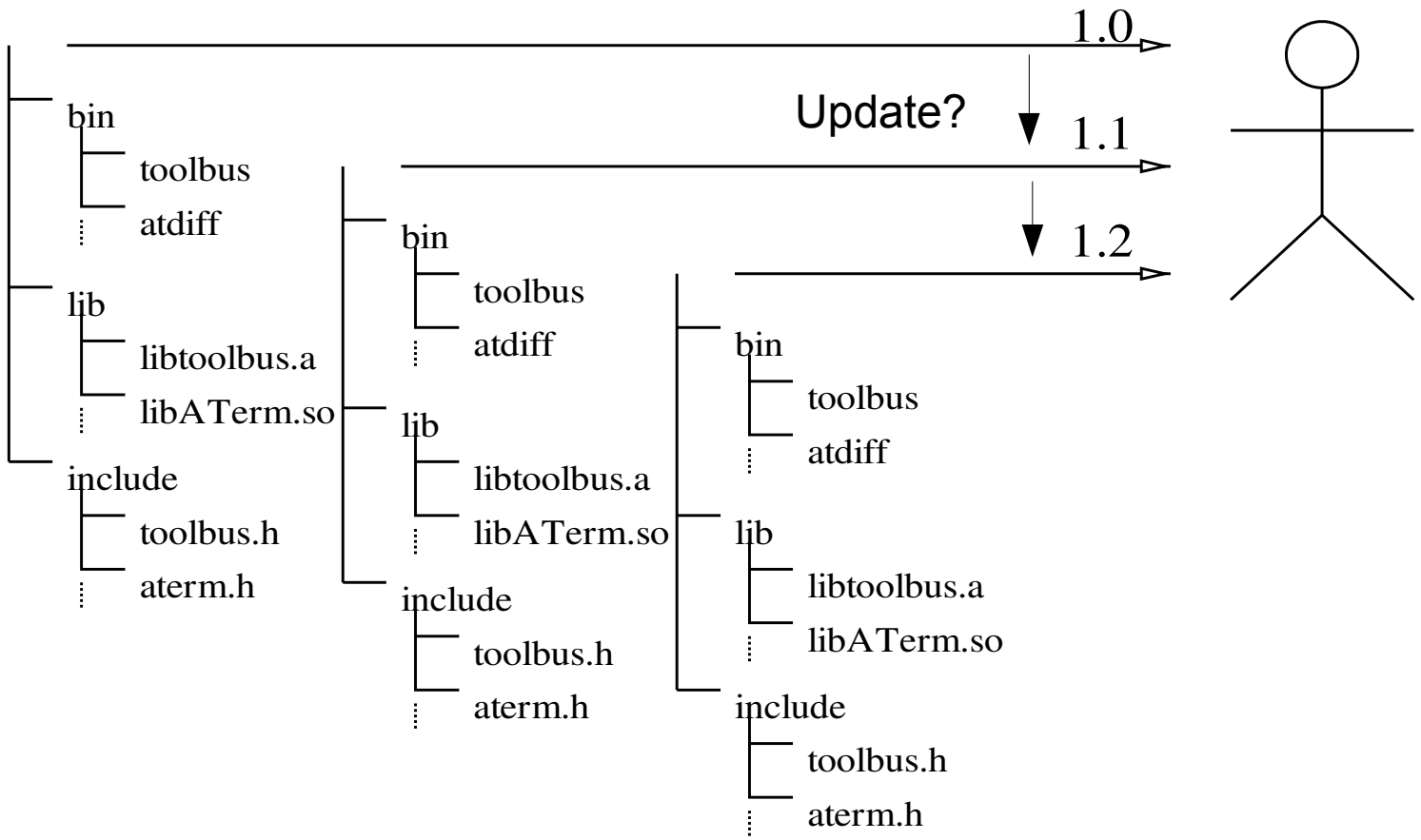


Then: `make, make check, make install`

Step 2: release compositions



Step 3: deliver to user



Challenges for continuous updates

- Space inefficient
  - Lot of duplication
- Time-consuming
  - Manual deployment
  - Bandwidth wasted
- Error-prone
  - No automatic undo
  - Traceability maintained manually

Binary Change Set Composition

- Solution:
  - Store binary files differentially
  - Use shallow copying for composition
  - Derive *composite changesets*
  - Update by transferring such changesets or their inverse
- Implementation:
  - Subversion

BCSC on top of Subversion

- BCSC maps to Subversion features:
  - Component binaries are *checked in*
  - Compositions created by *branching*
  - First user deployment: *checkout*
  - Upgrade/downgrade: *workspace switch*
- Additional benefits:
  - Traceability & transactions
  - *Branch* is constant space
  - *Switch* proportional in size of changeset

Evaluation

- Drawbacks of update facilities
  - Invasive (e.g. Nix, APT, RPM etc.)
  - Source-based (e.g. Ports)
  - Language dependent (e.g. JPloy)
- Binary Change Set Composition
  - Complexity is at the vendor side
  - Works with apps in binary form
  - Has no dependency on language or OS

Towards self-updating software

- BCSC requires no additional infrastructure on the user side
- In the case of Subversion
  - Application itself bundled with SVN *client*
  - Only first deployment is a manual step
  - “Update button” switches “workspace”
  - Goal: Mac OS X application bundles with efficient updates

Summary & Conclusions

- Binary change set compositions for efficiently updating heterogeneous component-based applications
  - Light-weight
  - Efficient
  - Safe
- Step towards self-updating software
- Future work: real case-study

Thank you

Questions?