

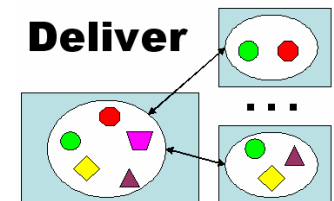
# Binary Change Set Composition

---

Tijs van der Storm

Centrum voor Wiskunde en Informatica

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 (BCSC)**
-

# Perspective: continuous delivery

---

- Continuous...
    - Integration
    - Release
    - Updates (this talk)
  - Setting: heterogeneous component-based applications
  - Assumption: automation is key
  - Steps towards self-updating software
-

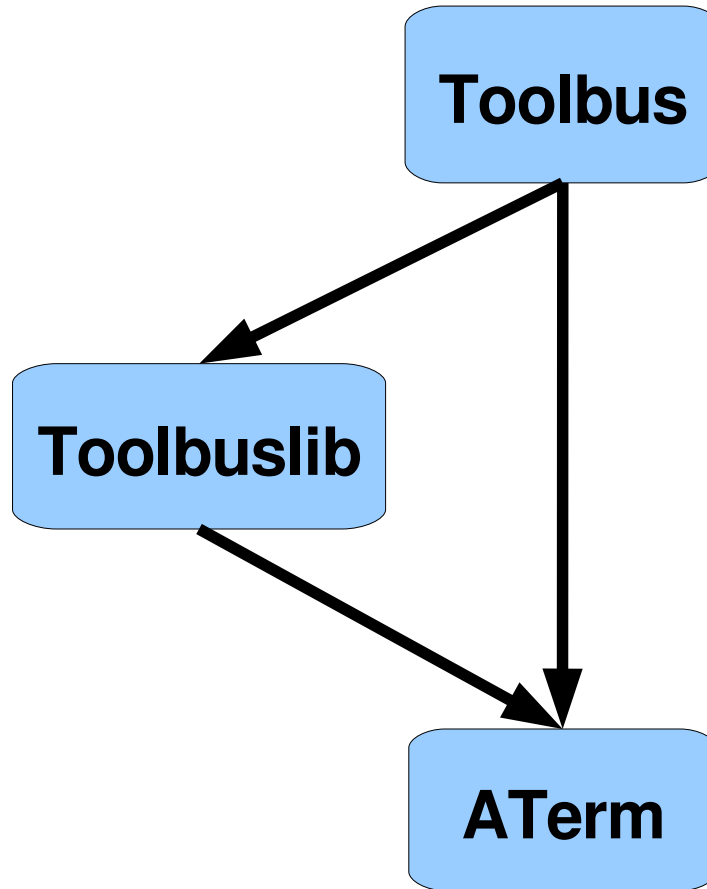
# Preliminaries

---

- Components:
    - *source trees with explicit dependencies*
  - Application:
    - *A closure of a component*
  - *Application release:*
    - *Building* each component
    - *Merging* all binaries of the closure
    - *Delivering* the result to users
-

# Example component-based application

---



# Step 1: building components

---

## □ Build in topological order:

```
./configure
```

```
--prefix=/install/toolbus
```

```
--with-toolbuslib=/install/toolbuslib
```

```
--with-aterm=/install/aterm
```

/install

├── toolbus

│ └── bin

├── toolbuslib

│ └── lib

│ include

└── aterm

│ └── lib

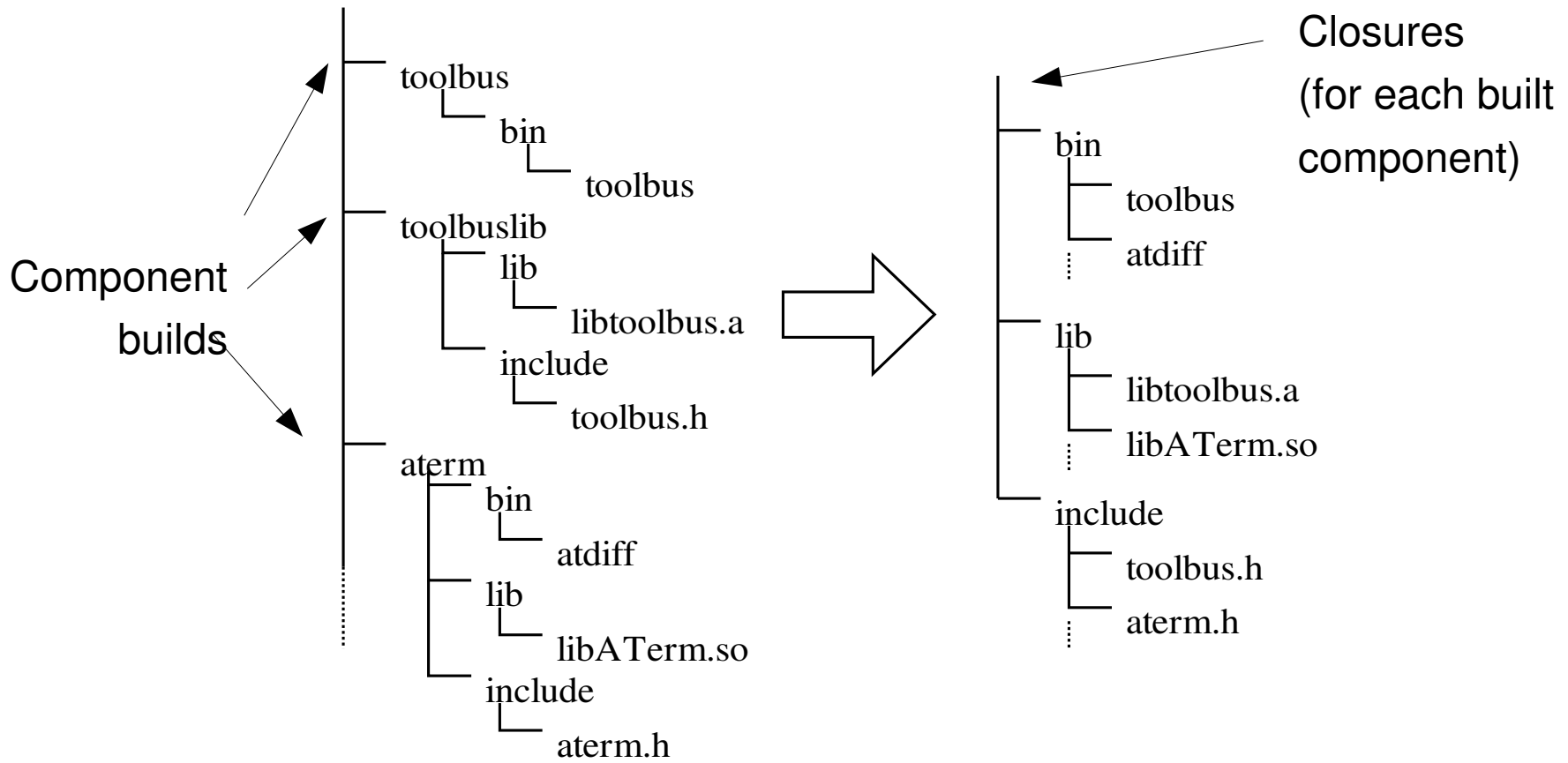
│ include

Then: make, make check, make install

---

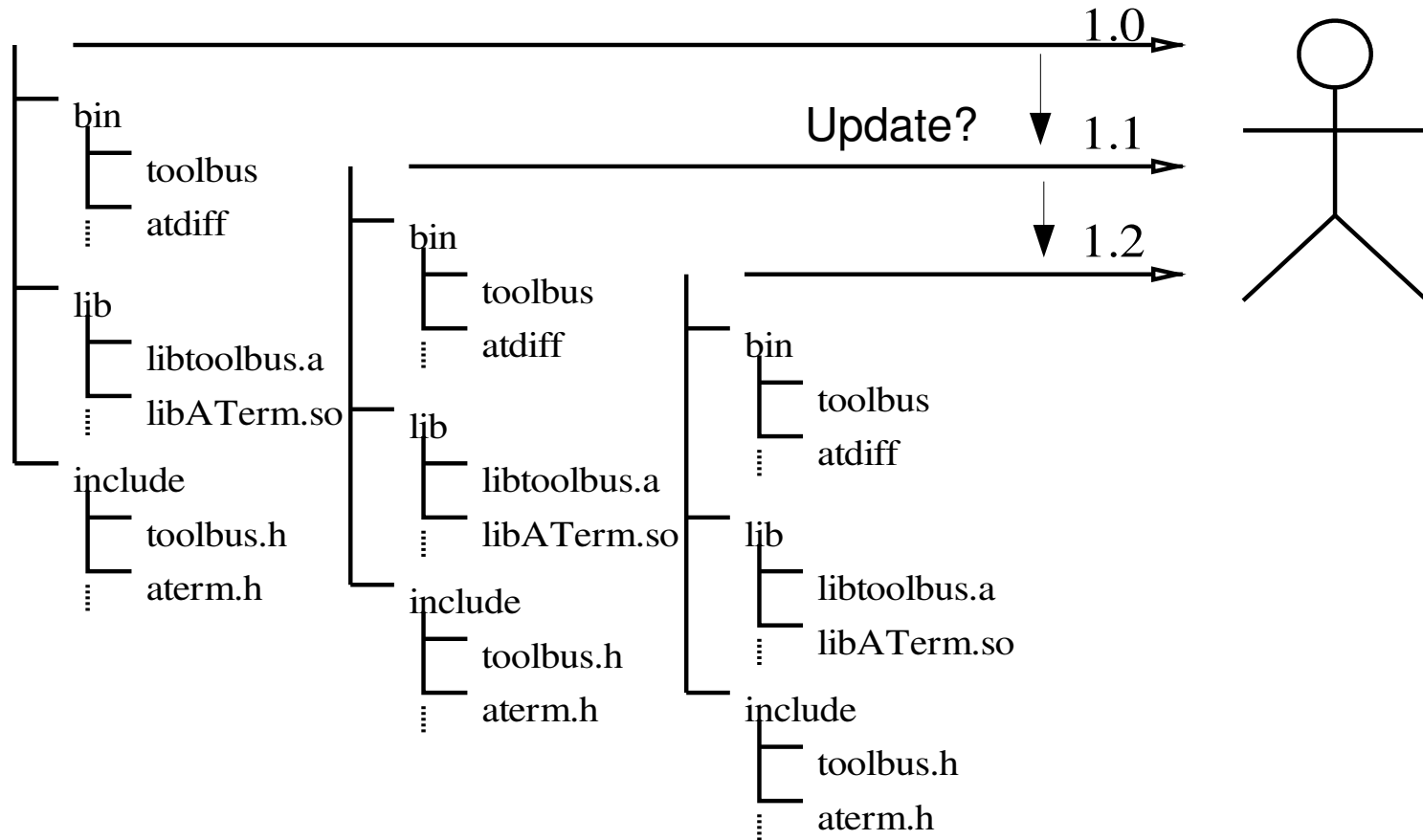
# Step 2: merging binaries

---



# 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* to create closures
    - Derive *composite changesets*
    - Update by transferring such changesets
  - 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 binary release
    - Has no dependency on language or OS
-

# Summary & Conclusions

---

- Binary change set composition for efficiently updating heterogeneous component-based applications
    - Light-weight
    - Efficient
    - Safe
  - Step towards self-updating software
  - Future work: prototype -> production
-

# Thank you

---

Questions?

---