Continuous Release and Upgrade of Component-Based Software

Tijs van der Storm

5th September 2005

Background

Project Deliver:

> *Automating software delivery through explicit knowledge management.*

Proposes Software Knowledge Based (SKB)

This talk:

- Automate release and upgrade of component-based software

Introduction

Motivation:

> *Frequent release in component-based setting is time-consuming and error-prone*

Automation of release is based on:

- VCS: components, revisions
- Integration test: acceptance predicate
- Dependency file: composition

Solution overview

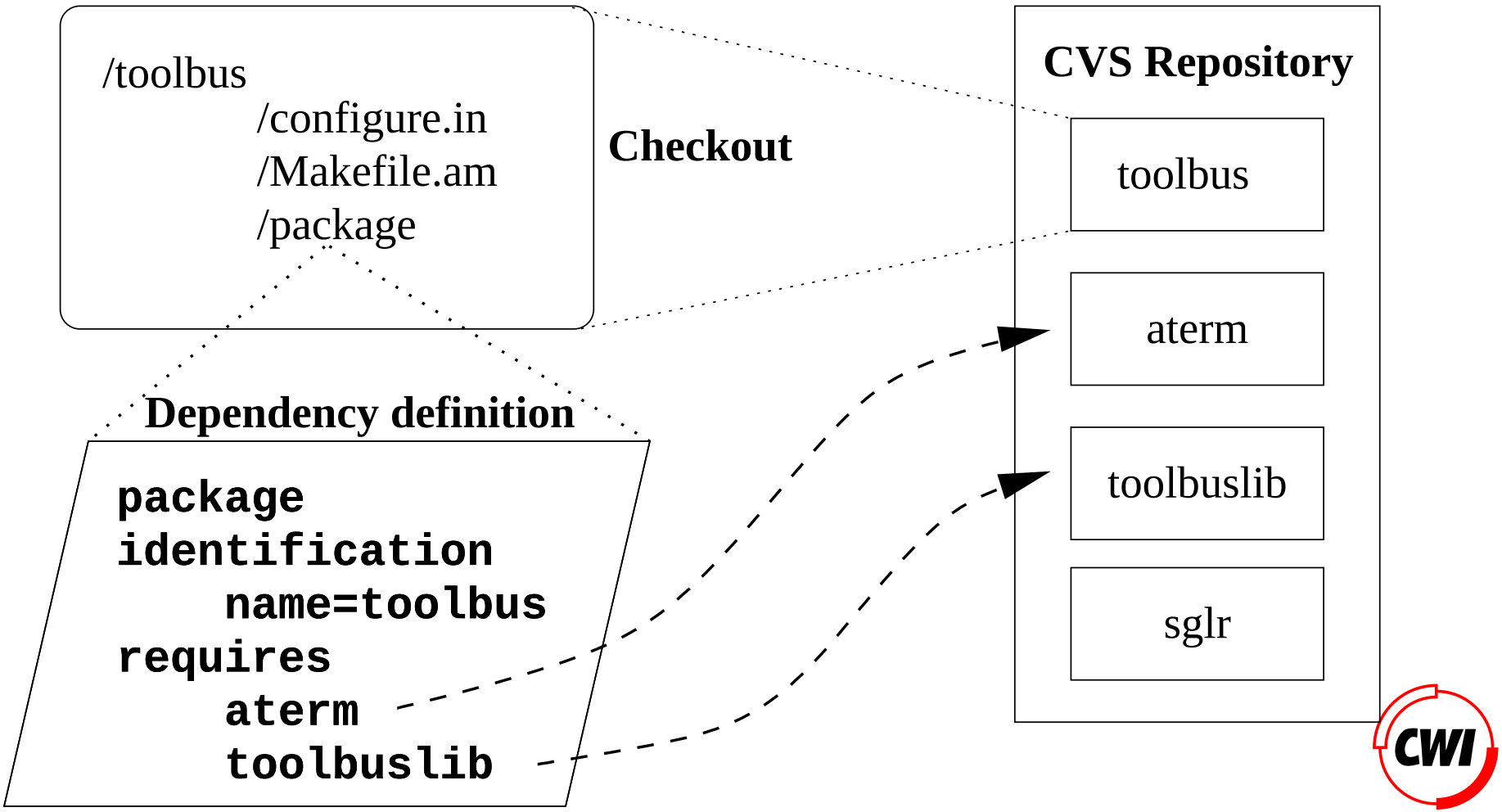
Extend continuous integration:

```
while true do
  foreach changed component do
    integrate
    if success then release
  end
end
```

Store results in SKB:

- derive updates from successful integrations
- maintain traceability through VCS revision ids

Setting


/toolbus
/configure.in
/Makefile.am
/package
Checkout
CVS Repository
toolbus
aterm
toolbuslib
sglr
Dependency definition
package
identification
name=toolbus
requires
aterm
toolbuslib

Model elements

Components:

- Component: module in VCS
- Revision: state of component at certain point in time

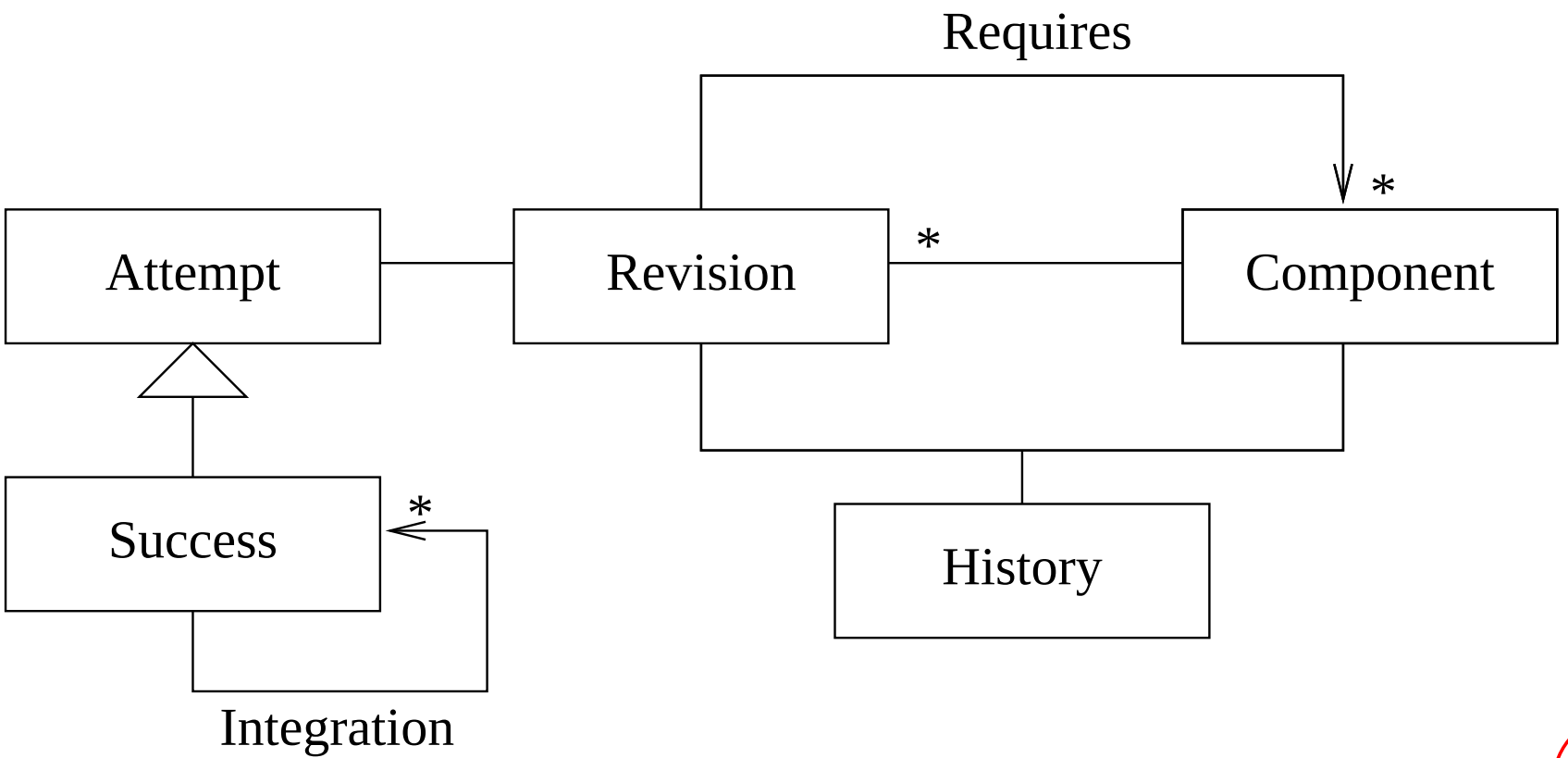
Integration:

- Successful and failed attempts

Dependencies:

- Declared dependencies (in source tree)
- Historic dependencies (at certain time)
- Integration dependencies (if integration success)

Component and evolution model



Relational Calculus

Formalisation using relational calculus

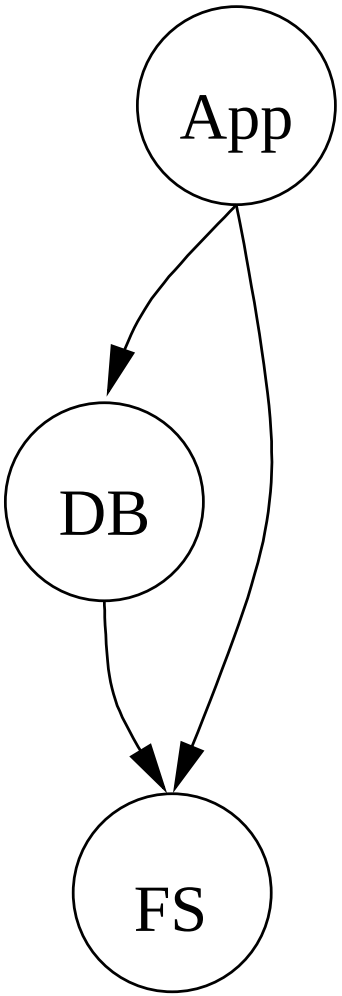
Two categories of sets/relations:

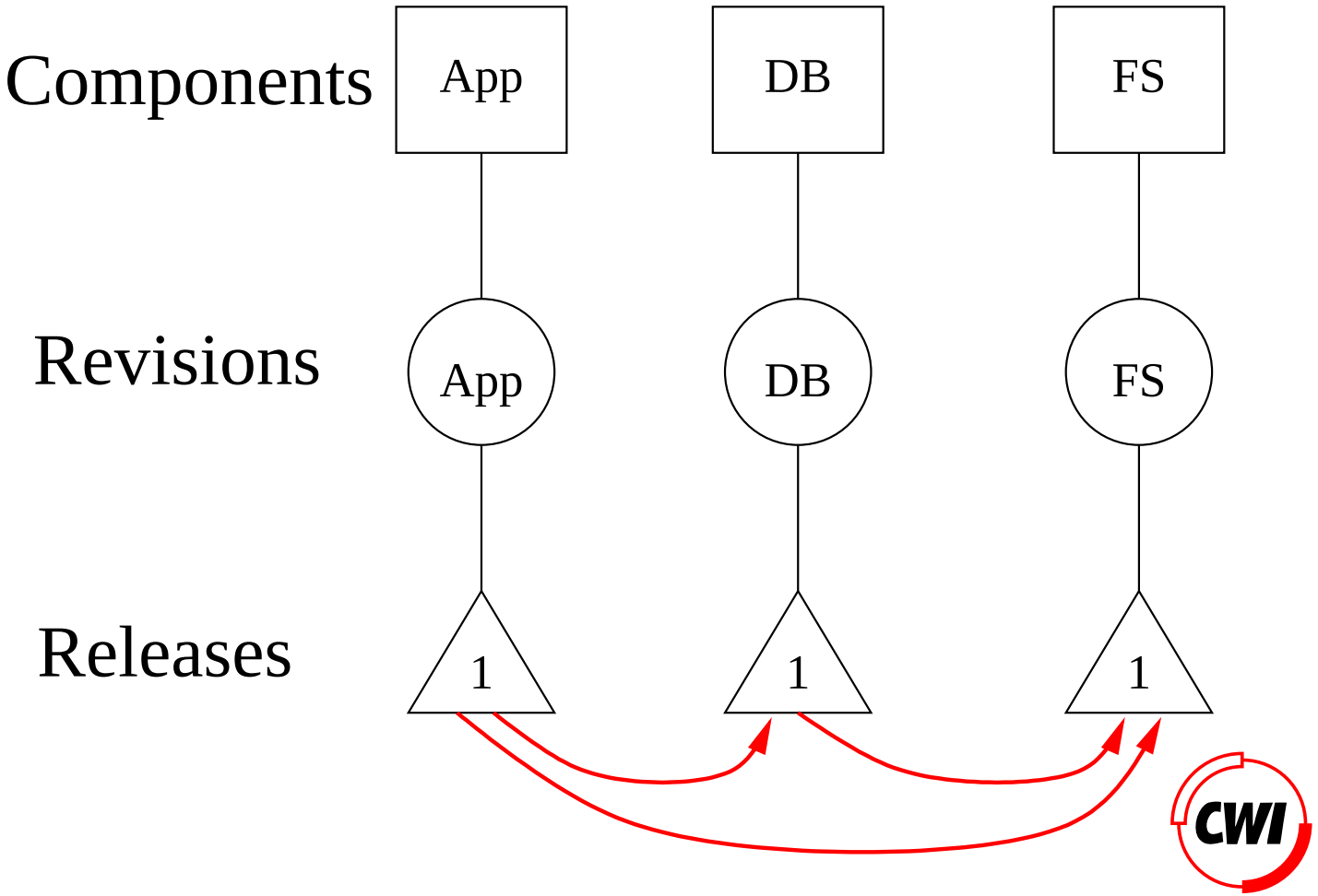
- Derived from VCS: *Component*, *Revision*, *Requires*
- Stored in knowledge base: *History*, *Attempt*, *Integration*

Relational expressions used to query the SKB

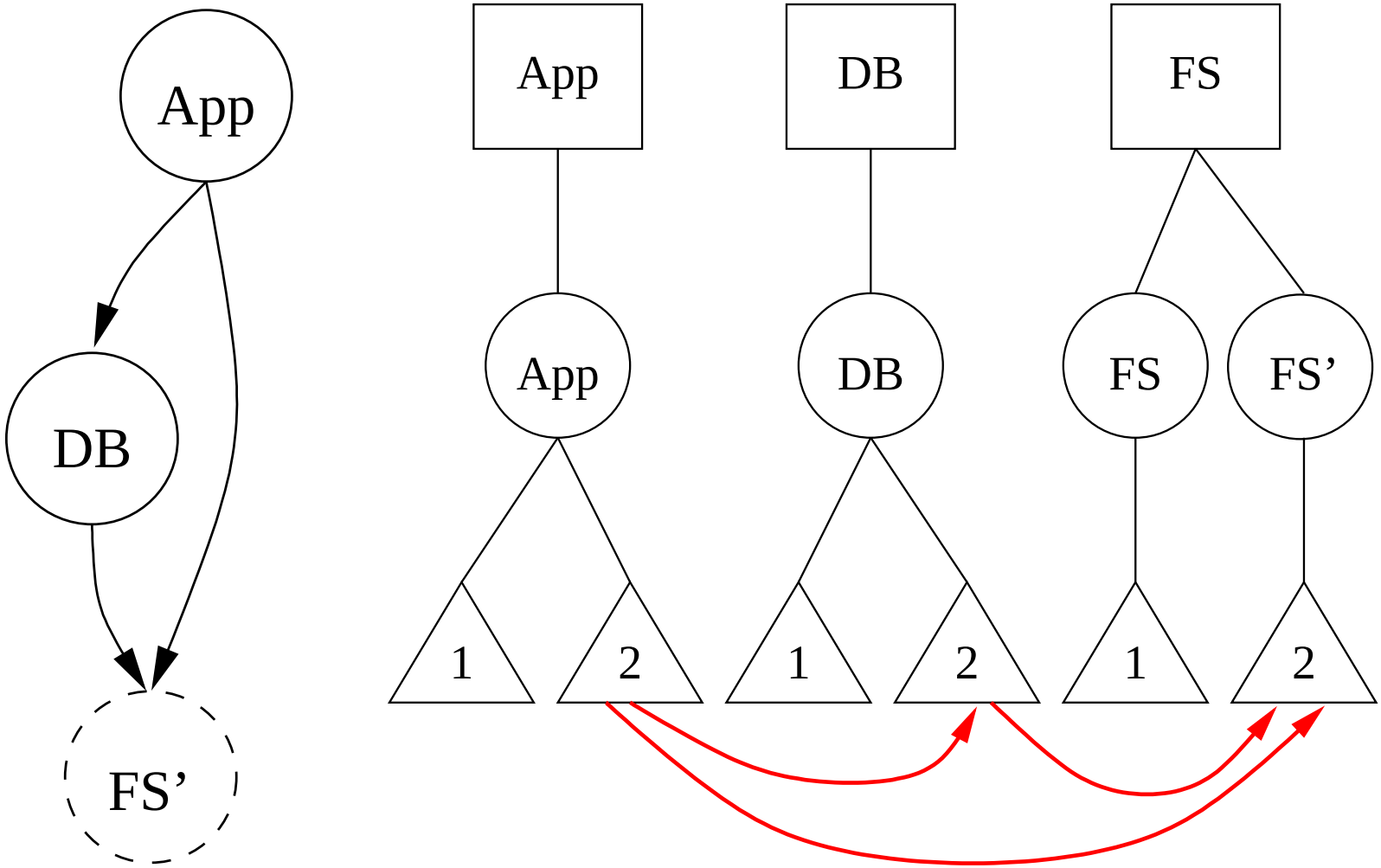
- $\setminus, \cup, \cap$
- image, domain, range, carrier etc.
- inverse, transitive closure

A first iteration

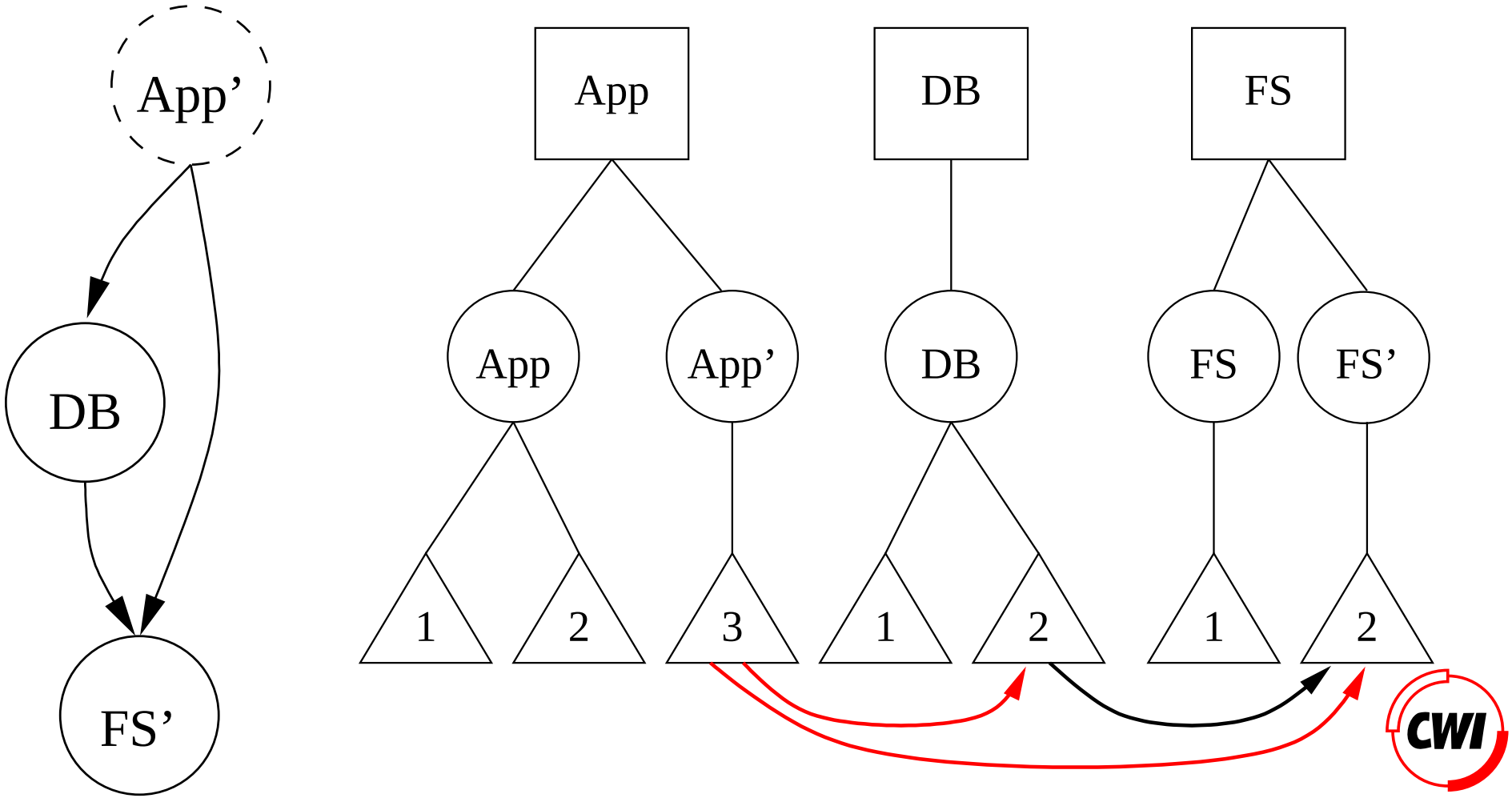




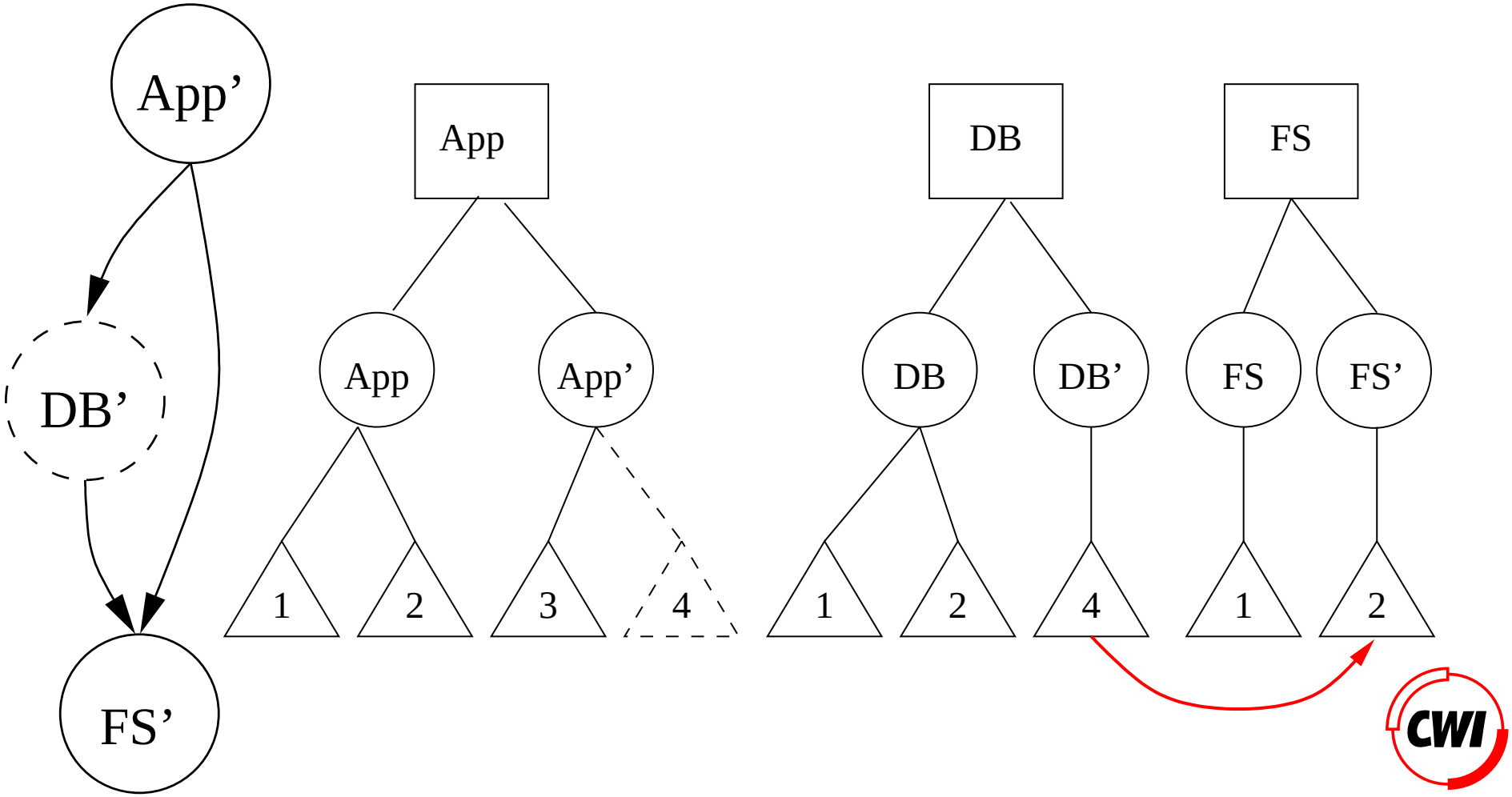
Change in the FS component



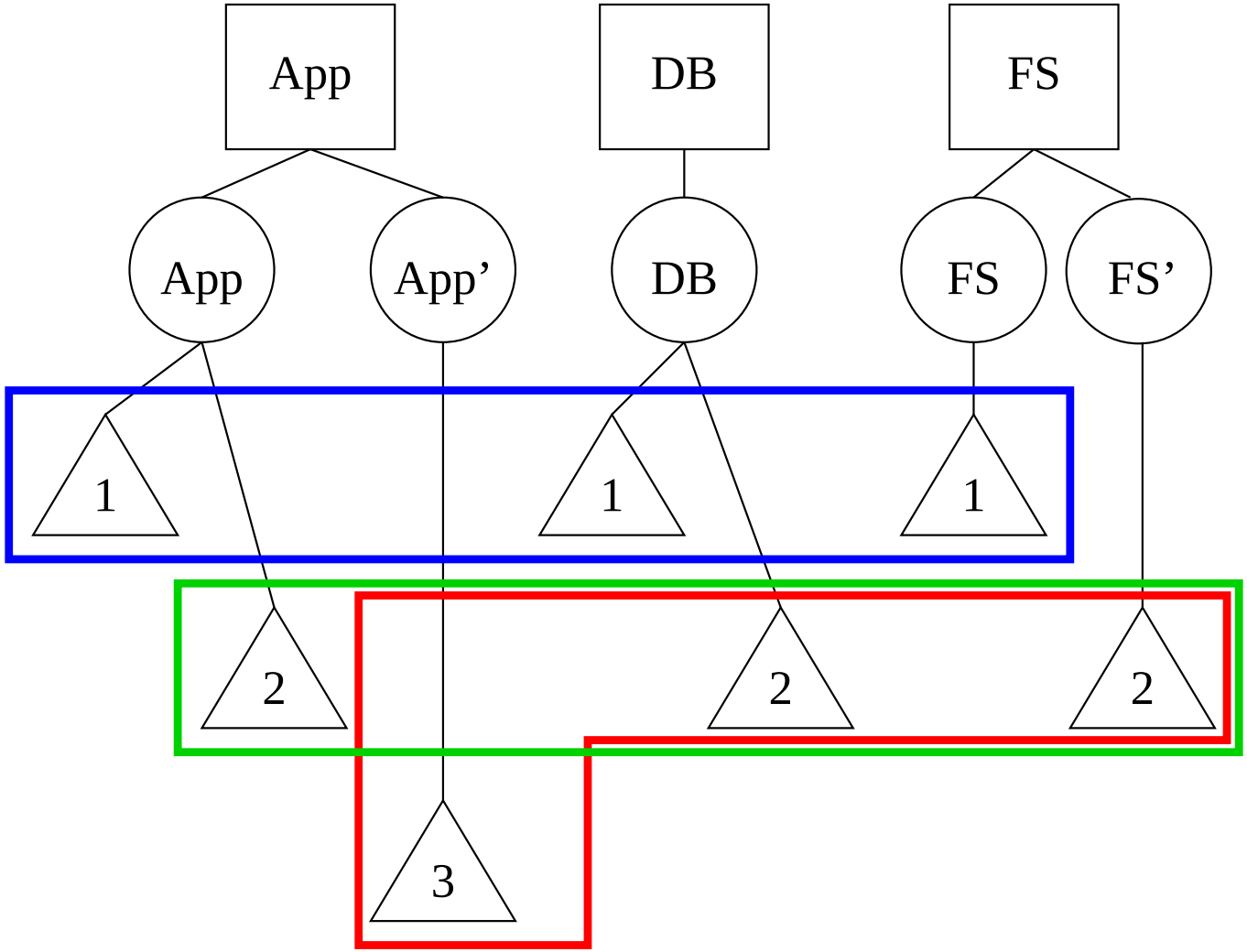
Change in the App component



Change in the DB component



Upgrading App



What to distribute

There may be many releases for one revision.

Reduce set of releases to singleton

- conservative, progressive
- ‘freeze’ installed component(s)
- (not) newer than

Bill of materials follows from integration dependencies.

Distribution is incremental wrt user configuration.

Evaluation

Advantages

- release is fully automatic and continuous
- updates are correct, complete and incremental

Disadvantages

- only one code line
- no support for build-time configuration
- no staging (e.g. alpha, beta, etc.)

Summary & future work

Summary:

- Continuous release based on continuous integration
- Formalisation of this domain in a component model
- Integrated components are released as stored in SKB
- Upgrades derived from SKB

Future work:

- Variants (branches)
- Build time configuration
- Introduce connection model
- Investigate datamining opportunities

End

- Deliver project: `http://www.cwi.nl/projects/deliver`
- More info: `http://www.cwi.nl/~storm`

Thank you!