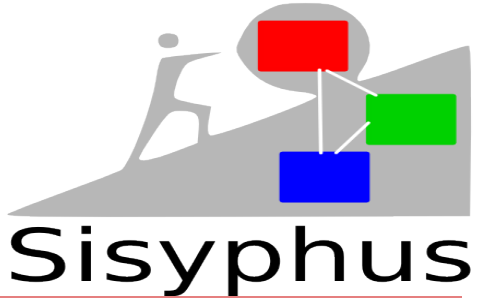
# Backtracking Incremental Continuous Integration

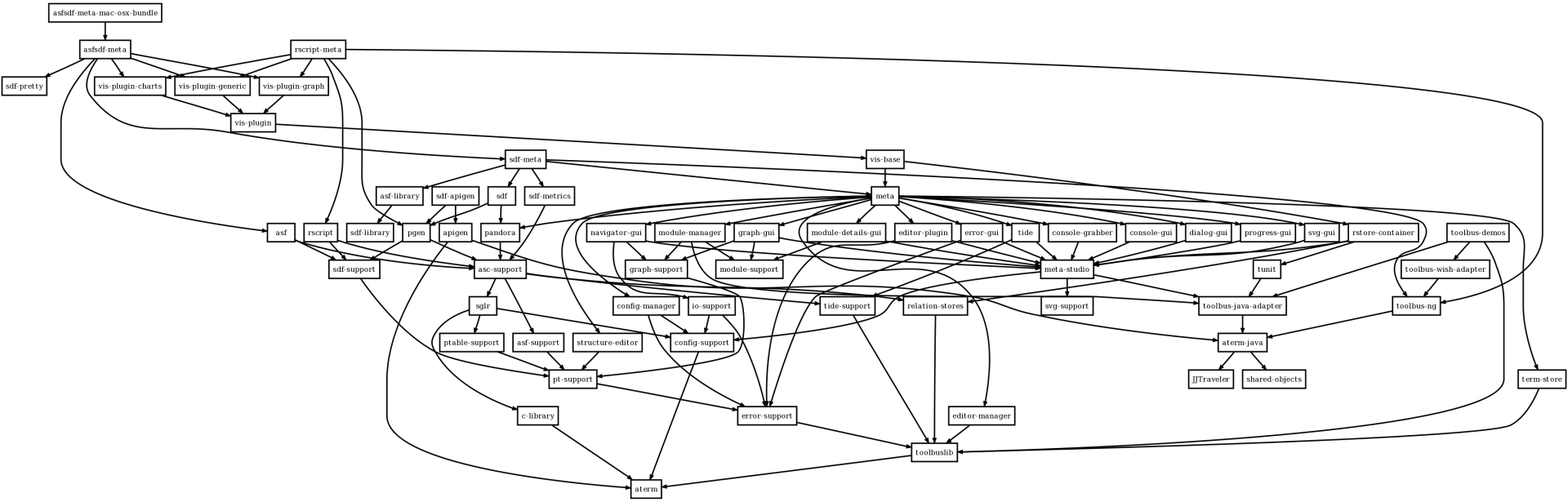
Tijs van der Storm
Centrum voor Wiskunde en Infromatica
storm@cwi.nl

# Introduction

- Continuous Integration (CI)
  - Is all about feedback on the effect of changes
  - *Heartbeat of software development*
- Today
  - How to improve feedback in large-scale component-based integration
  - Technique: backtracking incremental CI

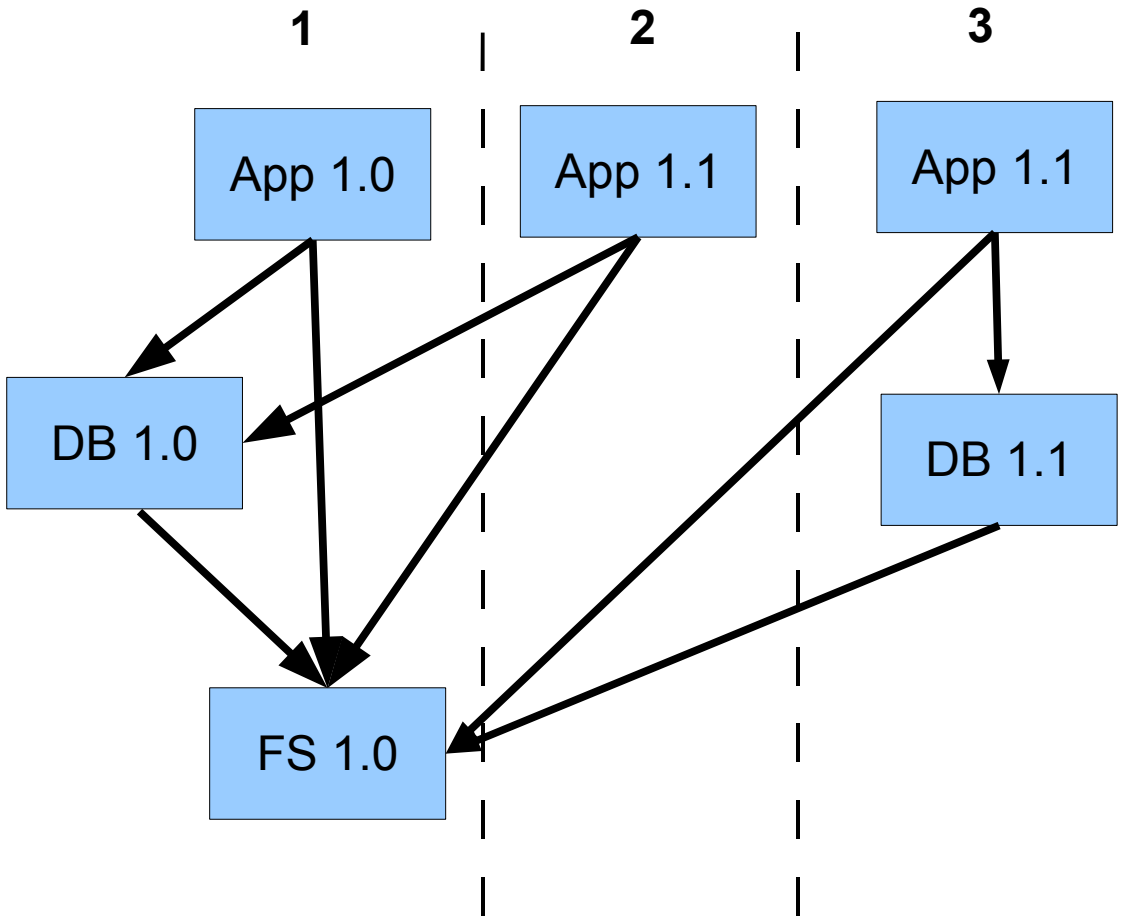
# Component-based integration



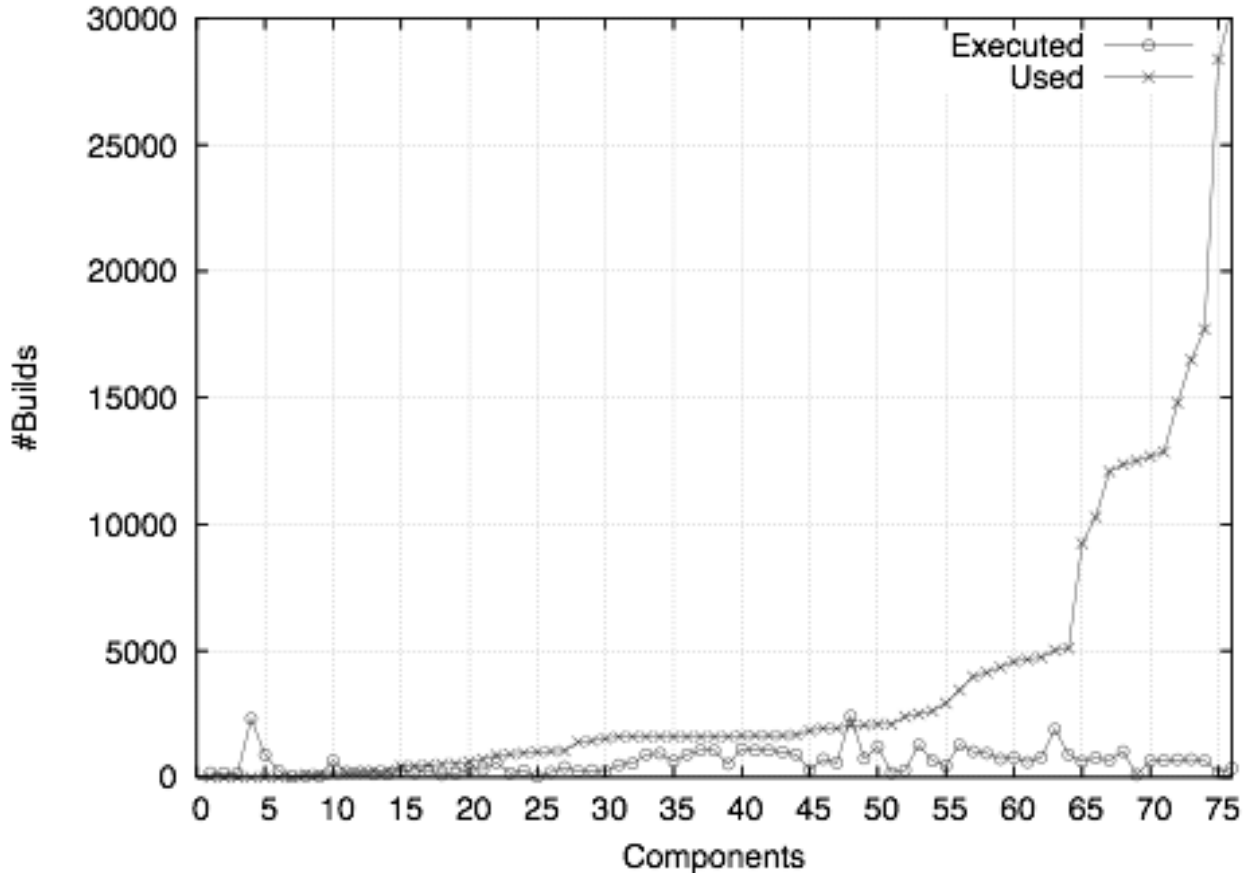
# Incremental integration

- Determine components affected by commit
  - components with changes
  - their co-dependencies
- Reuse earlier builds of unaffected components if needed
- Improves feedback by preventing redundant builds

# Incremental Integration



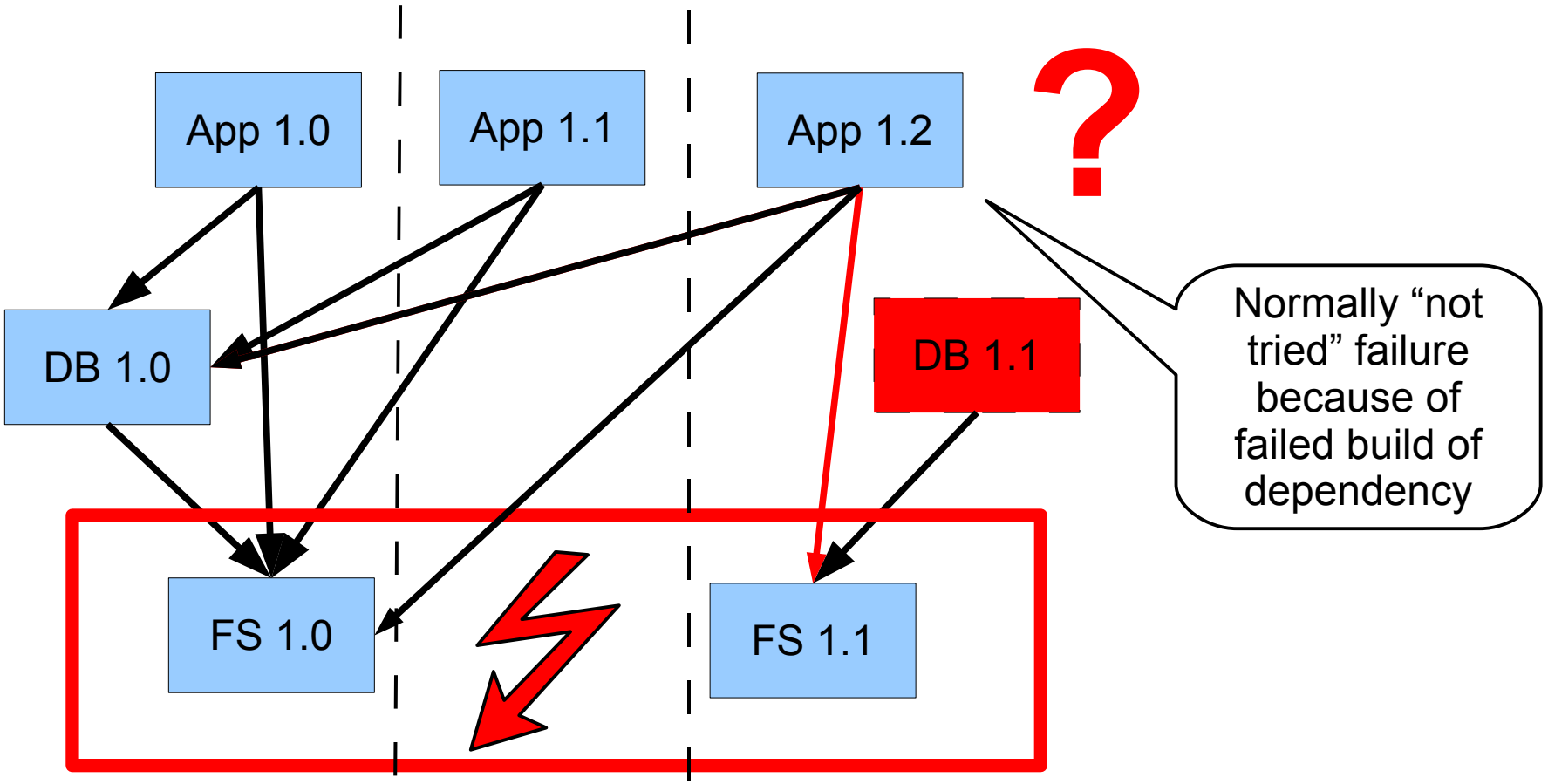
# Build reuse


Executed
Used
#Builds
Components
0
5000
10000
15000
20000
25000
30000
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75

# Build failures reduce feedback

- Build failure ripple effect
  - lower components that fail prevent **all** higher components to be built
  - result: lack of integration feedback
- "***A*** build is better than ***no*** build"
- Use "backtracking" to mitigate effect
  - improves feedback & release opportunity

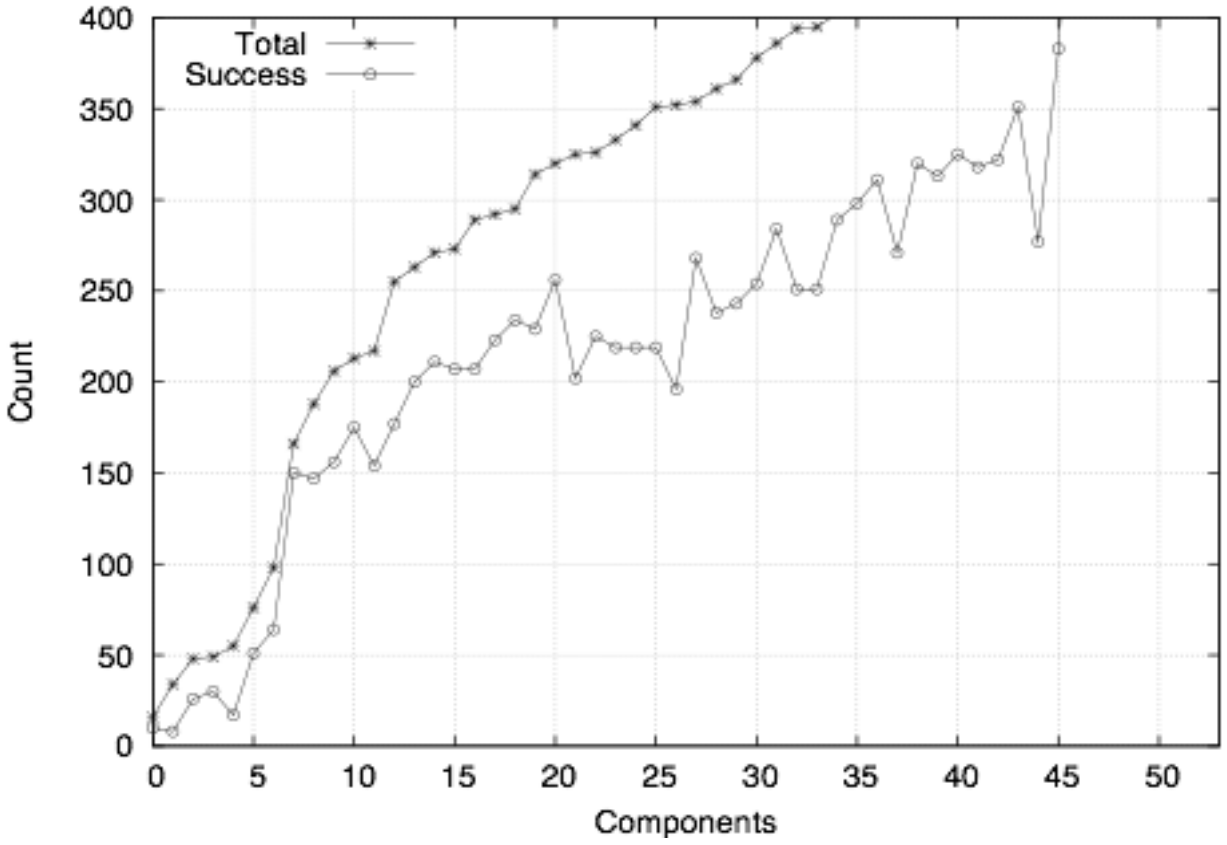
# Backtracking on build failure



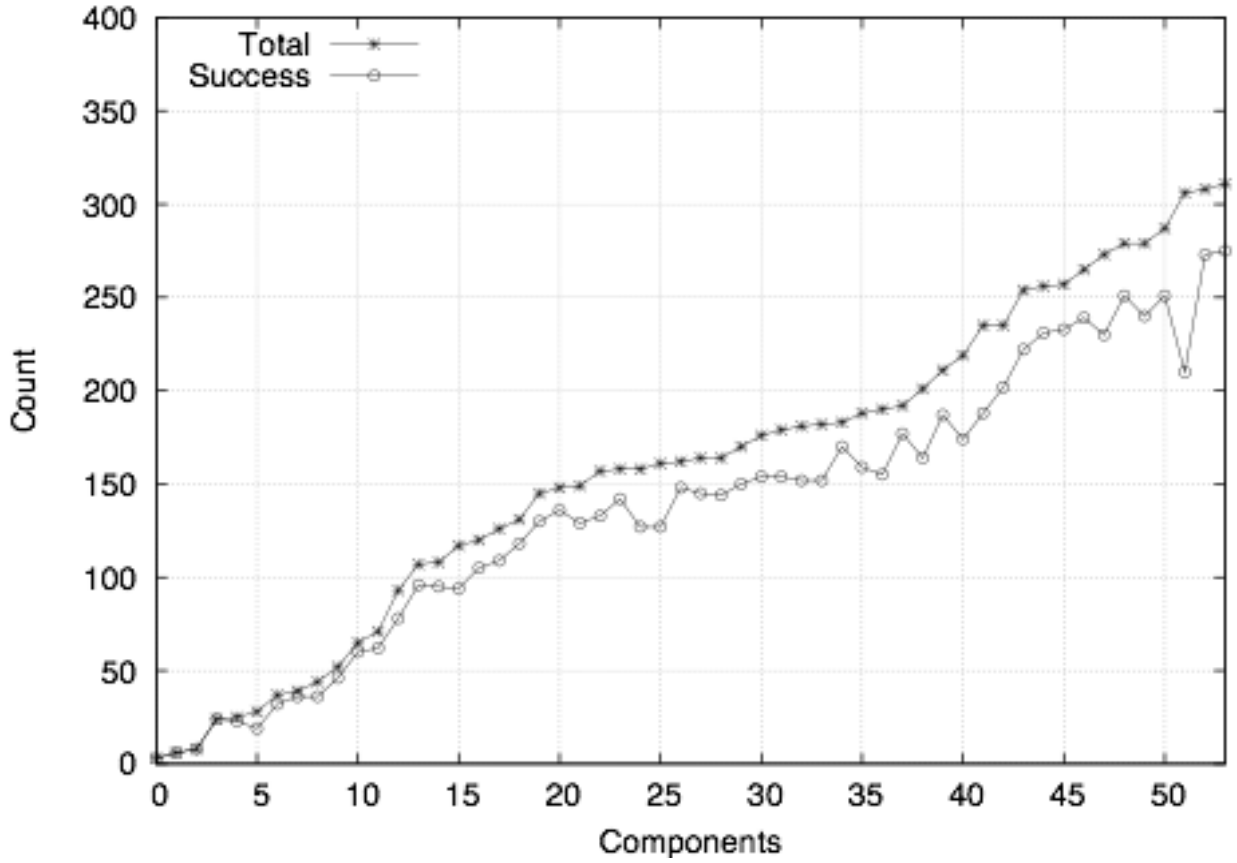
# Backtracking

- If a dependency build has failed
  - search for suitable sets of ***older*** builds
  - build against those dependencies
- Two types of backtracking:
  - “simple” (currently in use)
  - “true” (prototype)
- Both improve feedback by preventing “not tried” builds

# No backtracking



# With backtracking (“simple”)



# Conclusion

- Continuous integration is all about *feedback*
- Feedback suffers from:
  - redundant builds
  - ripple effect of build failures
- Feedback can be improved using:
  - incremental integration
  - backtracking integration

# Thank you

Questions?