# continuous integration : flex

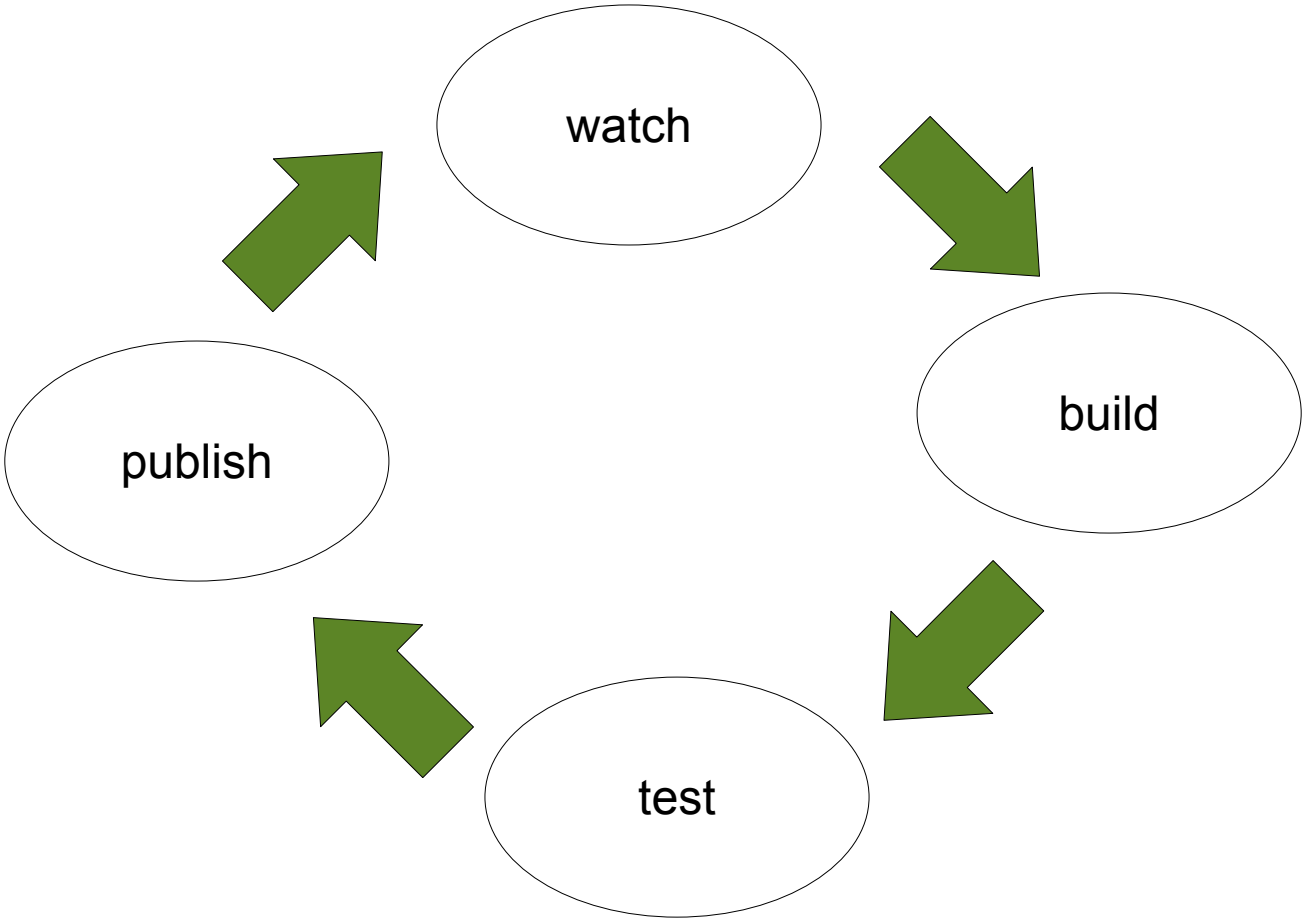
Brian LeGros
Adobe Developers of Greater Orlando

http://brianlegros.com/blog

# me

- application developer for ~9 years
- flex developer @ Highwinds CDN (http://www.highwinds.com)
- user group manager
  Rich ApplicatioN Developers of Melbourne(http://itsrandom.info)
  Adobe Developers of Greater Orlando (http://adogo.us)
- contributor to fluint project (http://code.google.com/p/fluint/)

# the continuous integration loop



# what?

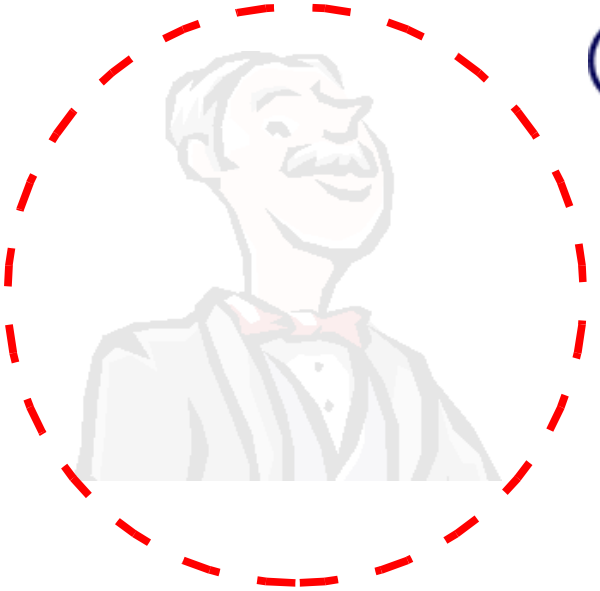
- watch source code for a change
- update copy of source code to build
- execute build script
- publish feedback to developers
- fix, if necessary, and repeat

# why?

- More feedback up-front rather than at release time
- Supplement communication in distributed teams
- Verify the deployment process in an integration environment
- On-demand artifacts for working, tested builds
- Eases the path to automated deployment
- Establishes confidence in the build process

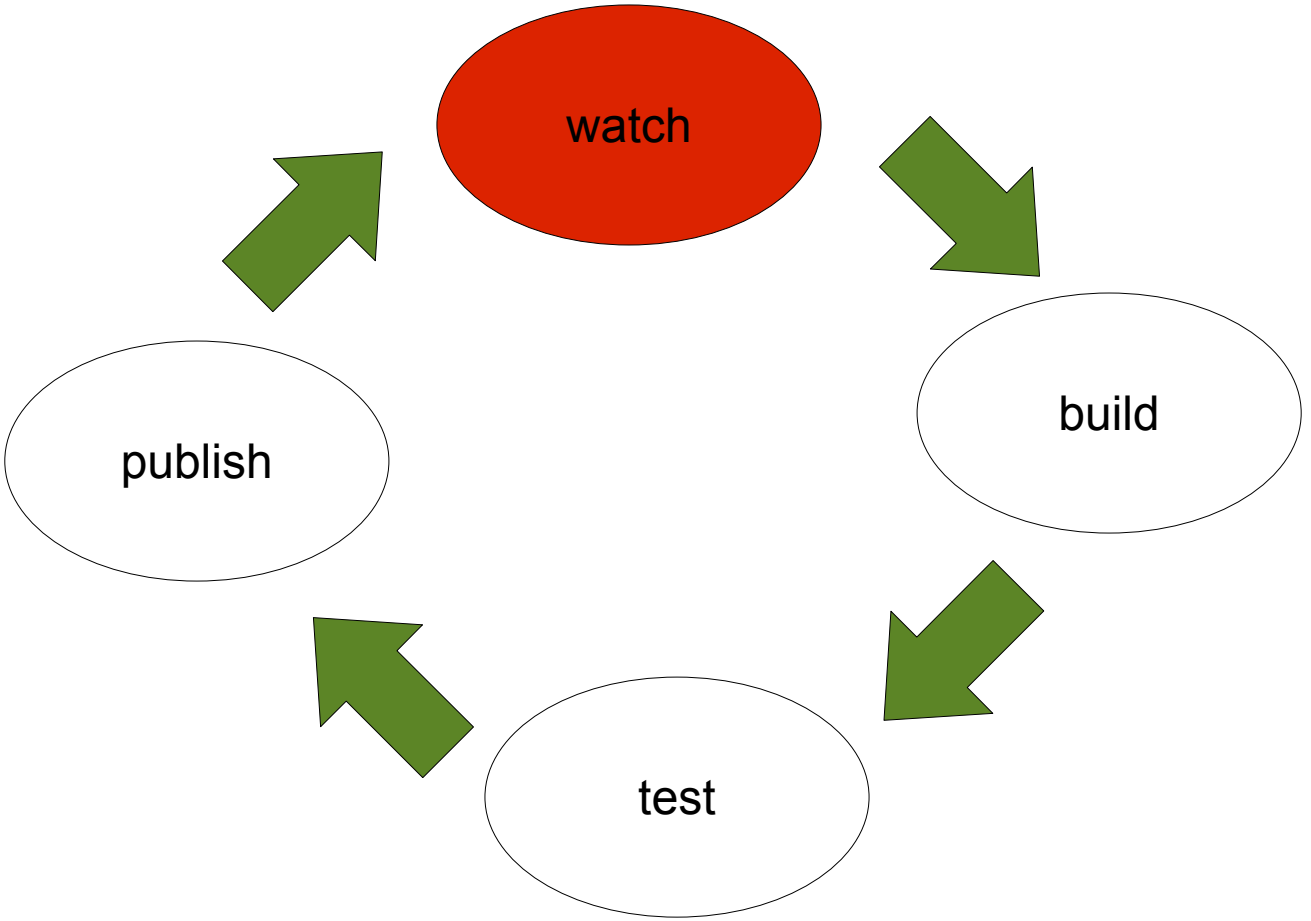
# options



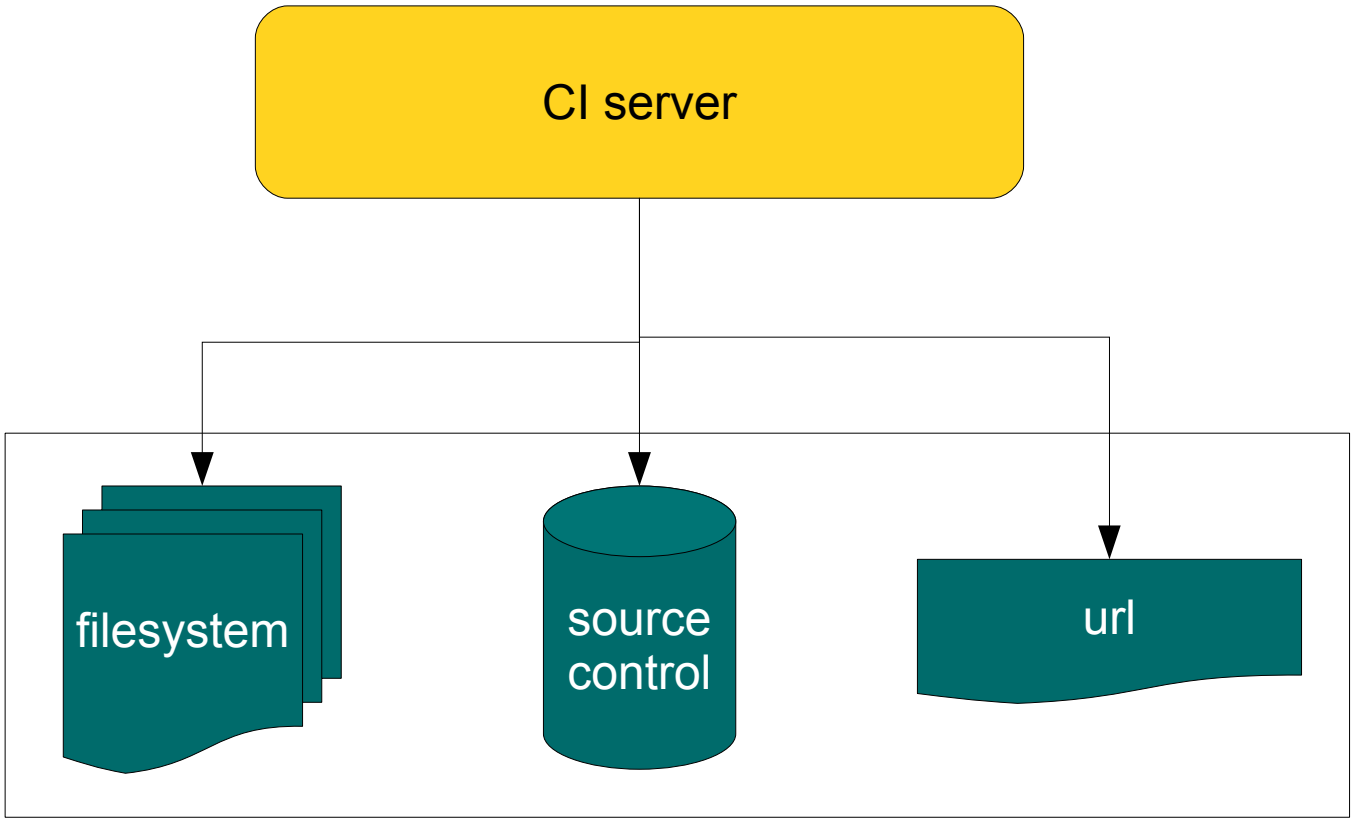




# the continuous integration loop



# watch



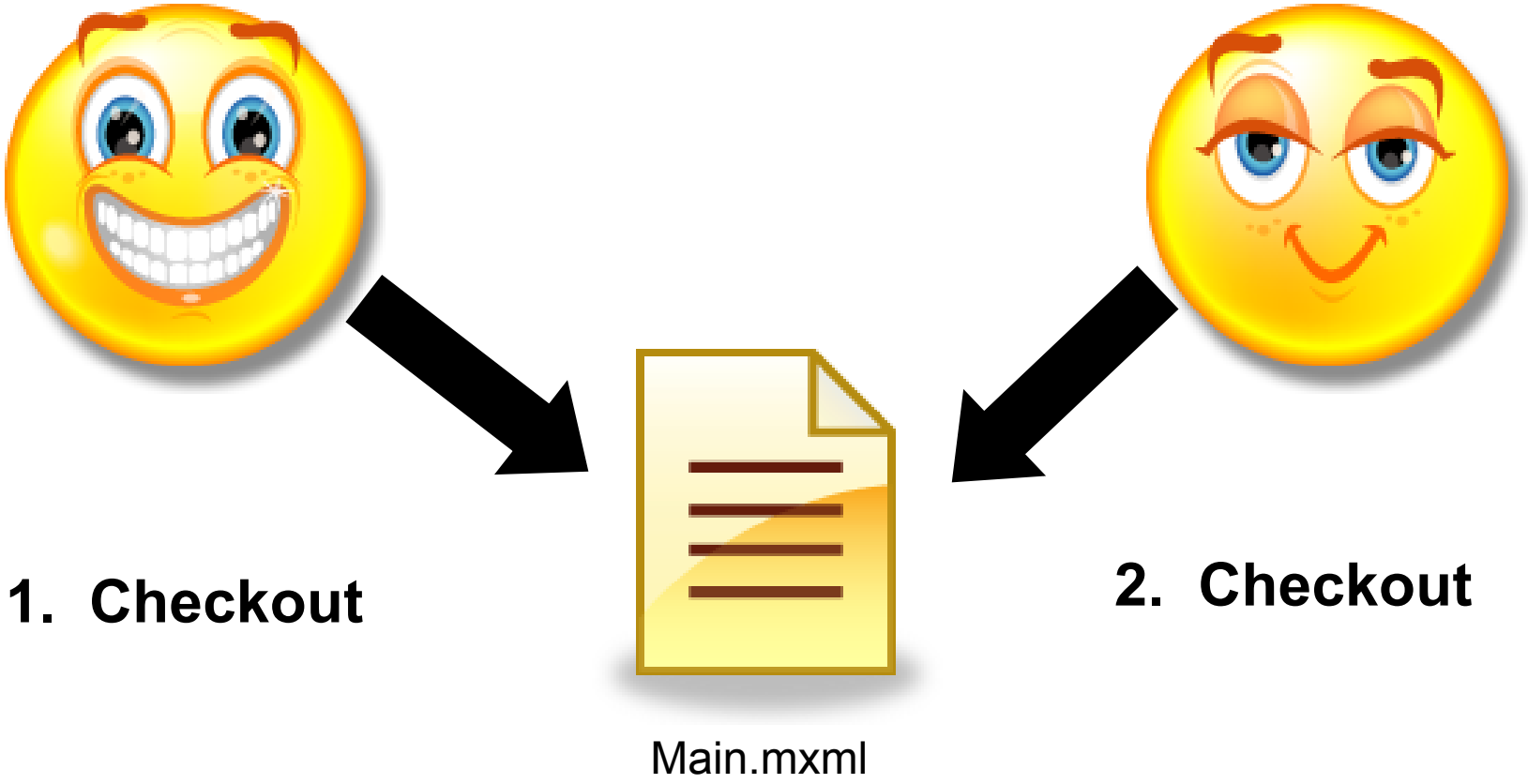
# source control considerations

- locking strategies
- folder management

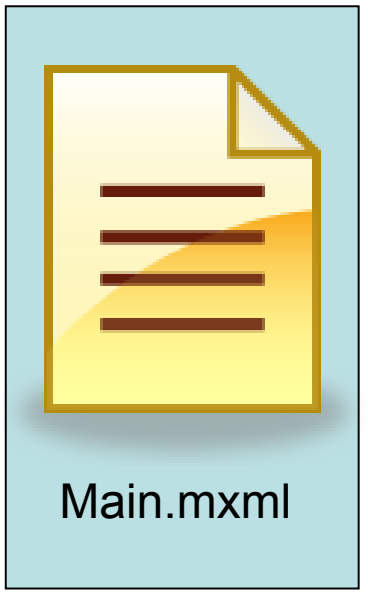
# pessimistic locking

Main.mxml

# pessimistic locking



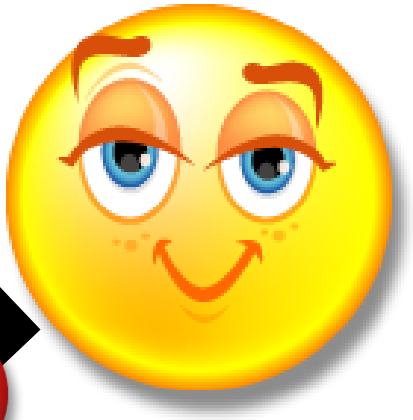
# pessimistic locking

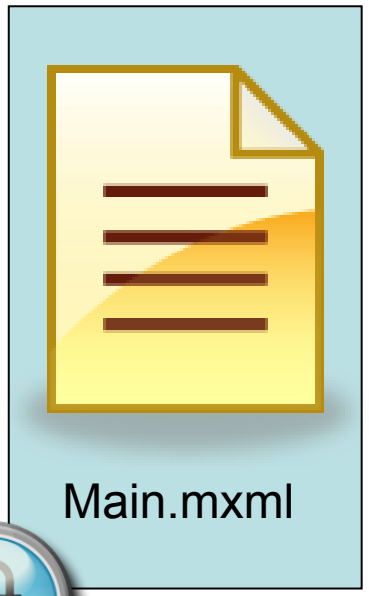



1. **Checkout**
3. **Lock**
4. **Edit**

2. **Checkout**

# pessimistic locking

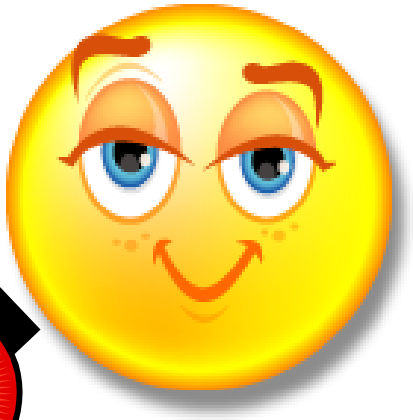


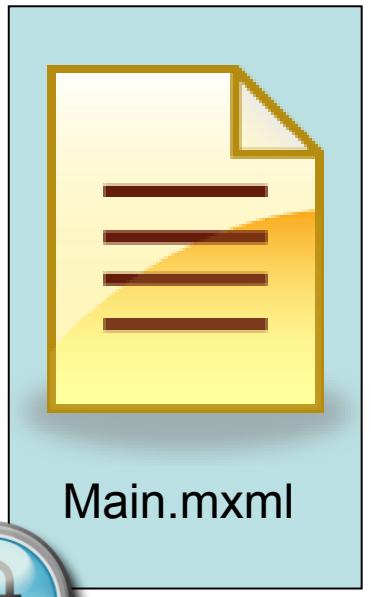



1. **Checkout**
3. **Lock**
4. **Edit**

2. **Checkout**
5. **Lock**

# pessimistic locking

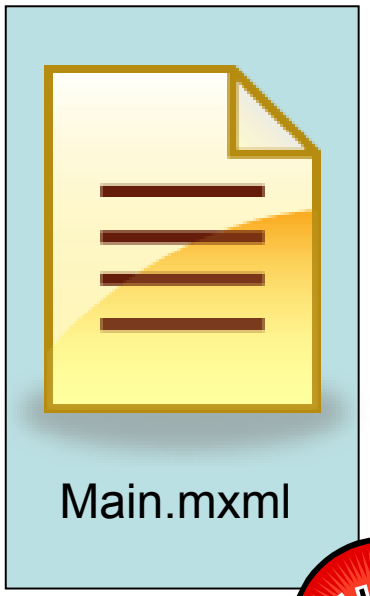





1. **Checkout**
3. **Lock**
4. **Edit**

2. **Checkout**
5. **Lock**
6. **Watch**

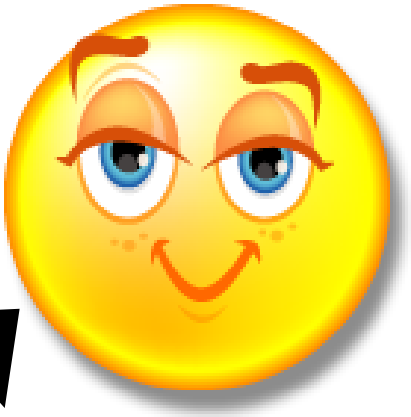
# pessimistic locking




1. **Checkout**
3. **Lock**
4. **Edit**
7. **Check-in**
8. **Unlock**

2. **Checkout**
5. **Lock**
6. **Watch**

# pessimistic locking

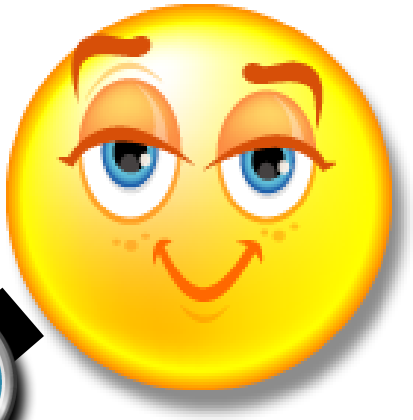


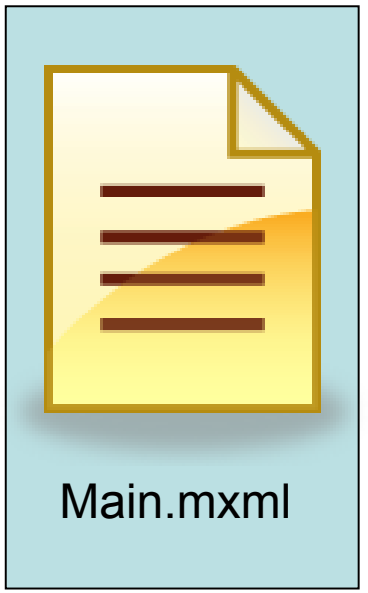
Main.mxml

1. **Checkout**
3. **Lock**
4. **Edit**
7. **Commit**
8. **Unlock**

2. **Checkout**
5. **Lock**
6. **Watch**
9. **Notify**

# pessimistic locking

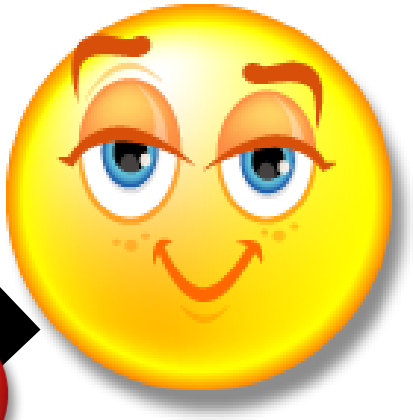


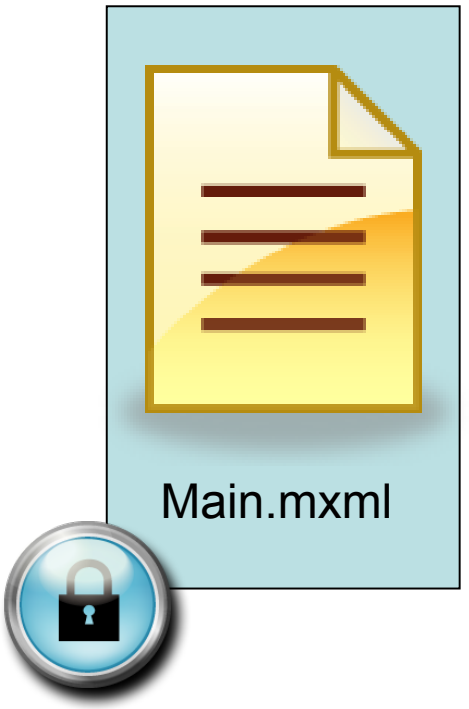



1. Checkout
3. Lock
4. Edit
7. Commit
8. Unlock

2. Checkout
5. Lock
6. Watch
9. Notify
10. Lock
11. Rinse/Repeat

# pessimistic locking - Highjack

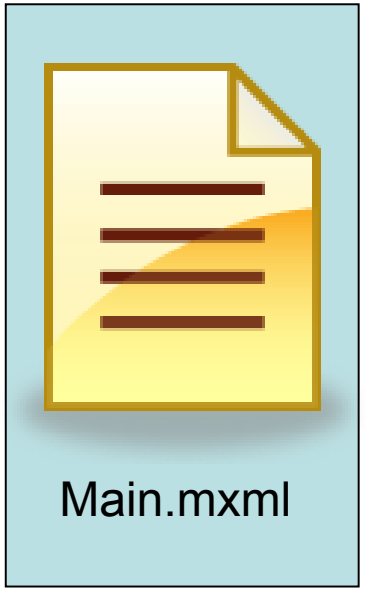





1. **Checkout**
3. **Lock**
4. **Edit**

2. **Checkout**
5. **Lock**

# pessimistic locking - Highjack




1. **Checkout**
3. **Lock**
4. **Edit**

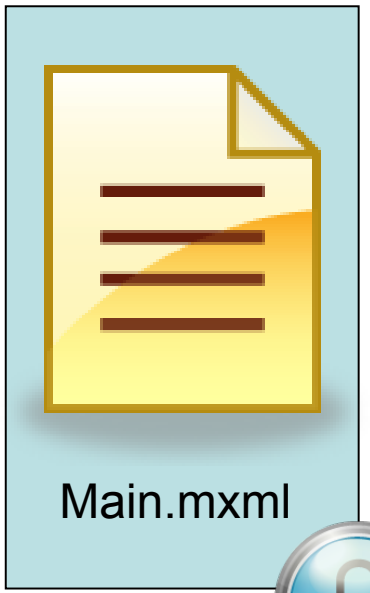
2. **Checkout**
5. **Lock**
6. **Hijack**

# pessimistic locking - Highjack


1. **Checkout**
3. **Lock**
4. **Edit**

2. **Checkout**
5. **Lock**
6. **Hijack**
7. **Edit**

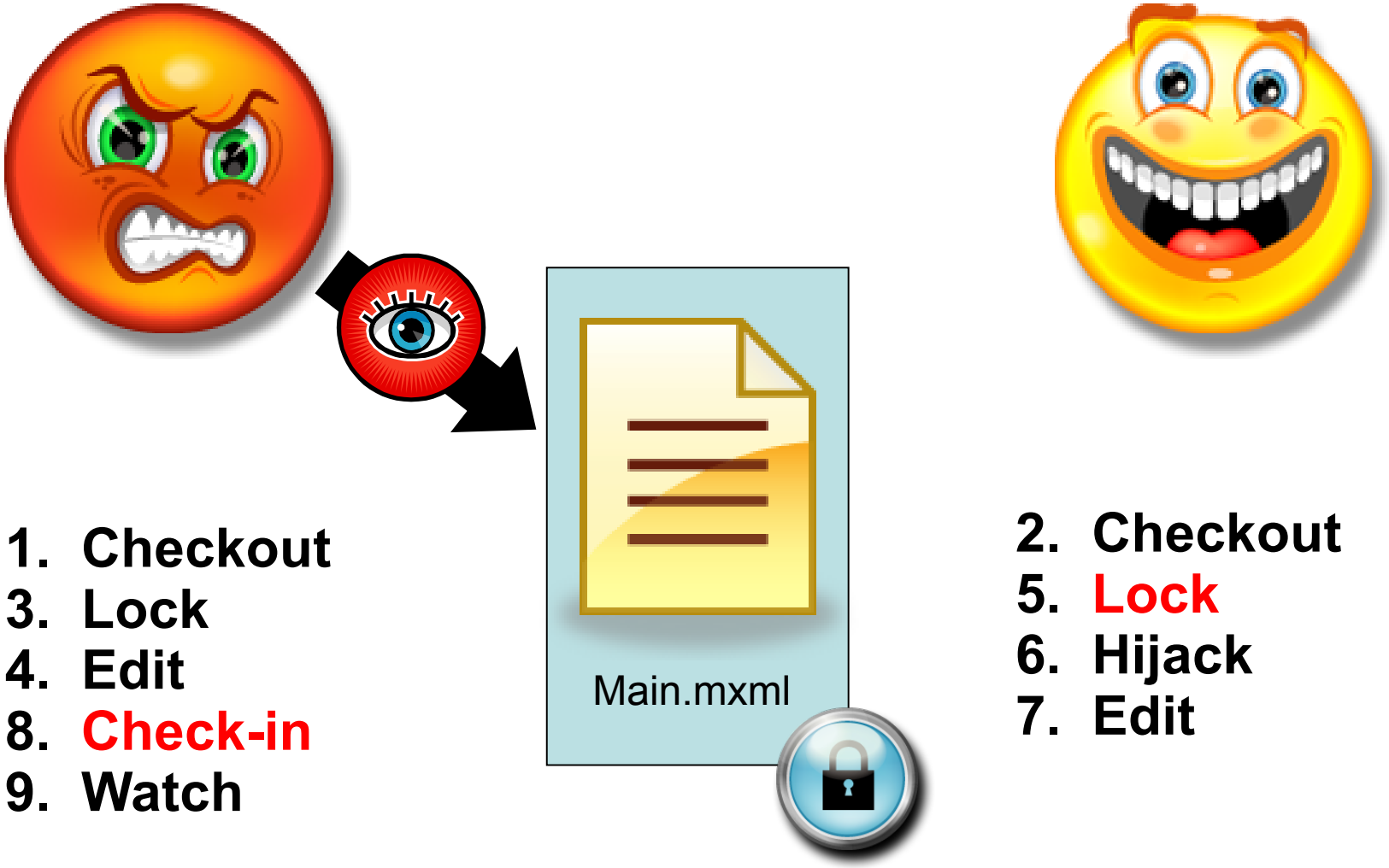
# pessimistic locking - Highjack




1. Checkout
3. Lock
4. Edit
8. Check-in

2. Checkout
5. Lock
6. Hijack
7. Edit

# pessimistic locking - Highjack




1. Checkout
3. Lock
4. Edit
8. Check-in
9. Watch

2. Checkout
5. Lock
6. Hijack
7. Edit

# pessimistic locking

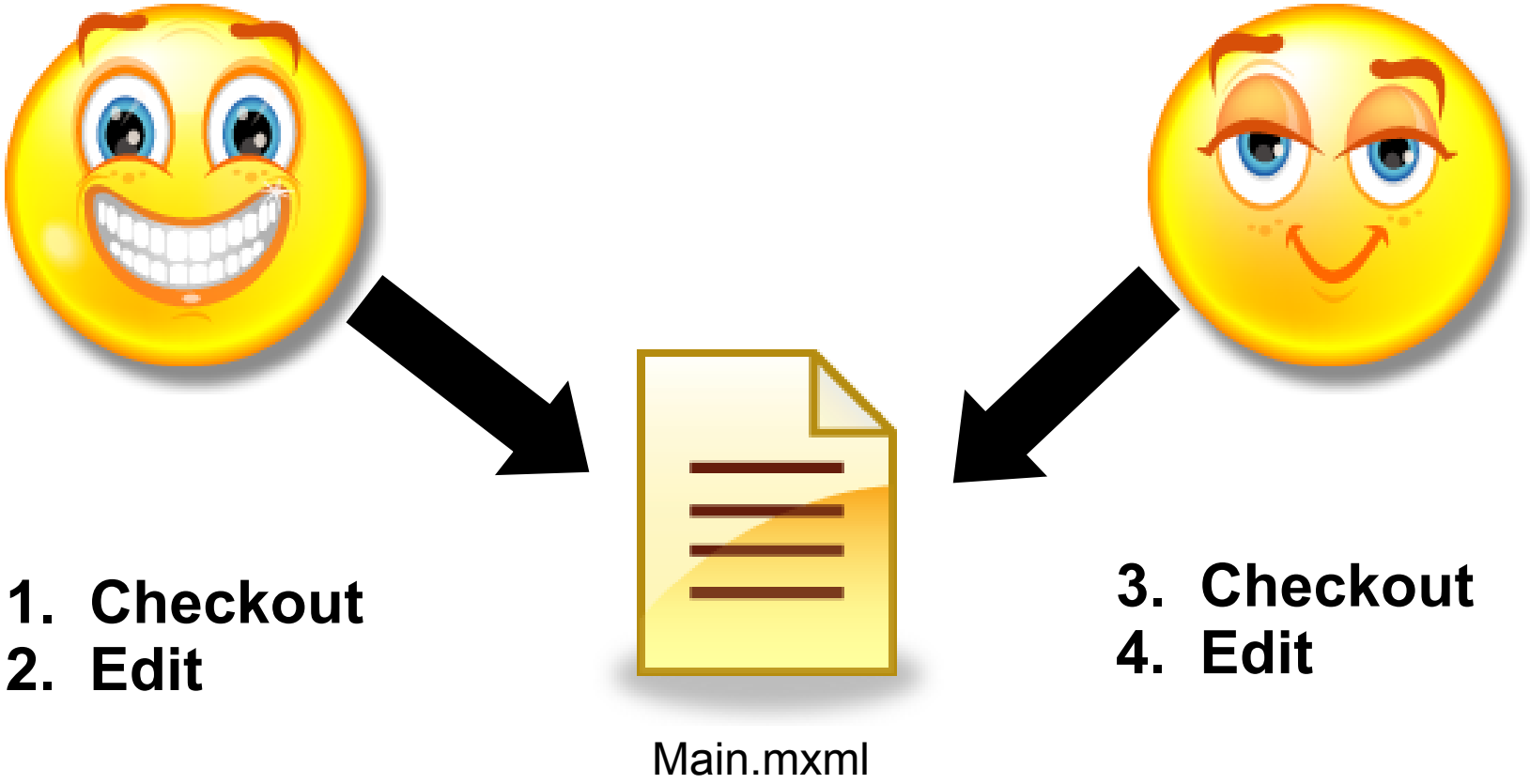
- Benefits
  - No conflict resolution needed
  - Small learning curve

- Challenges
  - Sequential development creep
  - Managing hijackers
  - May cause a CI build to fail if resources are locked

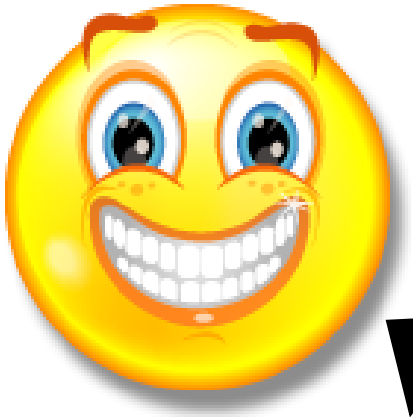
# optimistic locking

Main.mxml

# optimistic locking



# optimistic locking



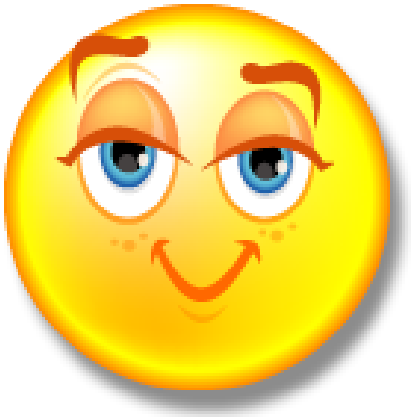
Main.mxml

1. **Checkout**
2. **Edit**
3. **Update**

3. **Checkout**
4. **Edit**

# optimistic locking

Main.mxml

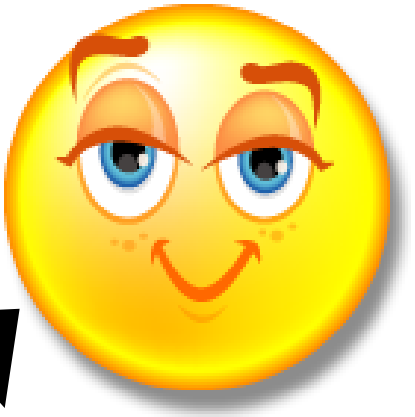


1. **Checkout**
2. **Edit**
3. **Update**
4. **Check-in**

3. **Checkout**
4. **Edit**

# optimistic locking

Main.mxml



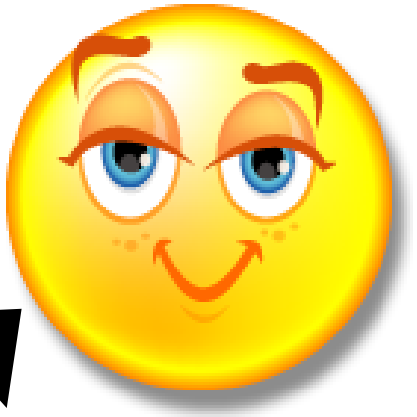
1. **Checkout**
2. **Edit**
3. **Update**
5. **Check-in**

3. **Checkout**
4. **Edit**
6. **Update**

# optimistic locking

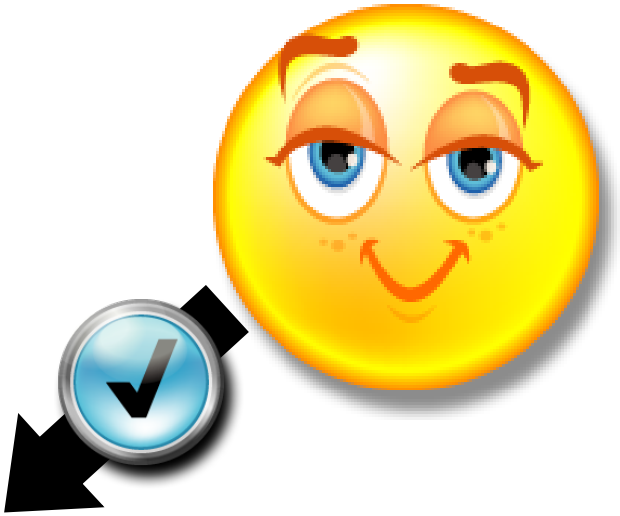
1. **Checkout**
2. **Edit**
3. **Update**
5. **Check-in**

Main.mxml



3. **Checkout**
4. **Edit**
6. **Update**
7. **Resolve**

# optimistic locking



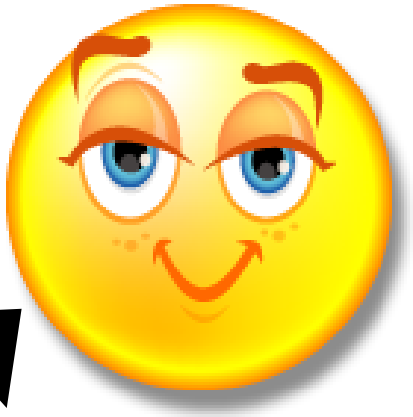
Main.mxml

1. **Checkout**
2. **Edit**
3. **Update**
5. **Check-in**

3. **Checkout**
4. **Edit**
6. **Update**
7. **Resolve**
8. **Check-in**

# optimistic locking - Overwrite

Main.mxml



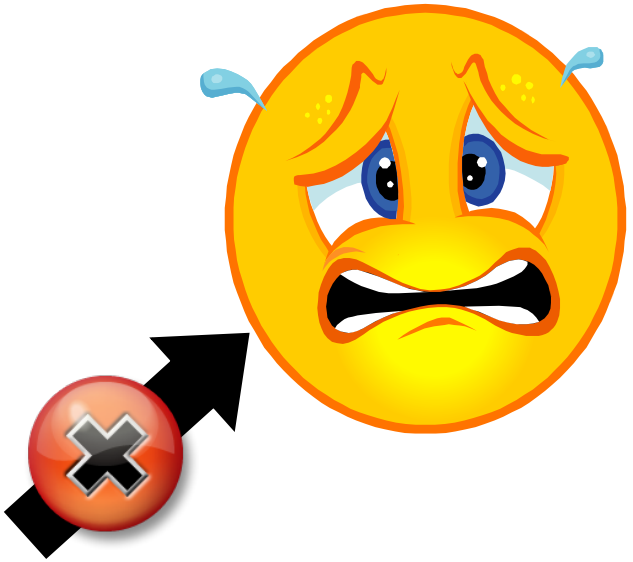
1. **Checkout**
2. **Edit**
3. **Update**
5. **Check-in**

3. **Checkout**
4. **Edit**
6. **Update**

# optimistic locking - Overwrite

1. **Checkout**
2. **Edit**
3. **Update**
5. **Check-in**

Main.mxml



3. **Checkout**
4. **Edit**
6. **Update**
7. **Ignore**

# optimistic locking - Overwrite

Main.mxml

1. **Checkout**
2. **Edit**
3. **Update**
5. **Check-in**

3. **Checkout**
4. **Edit**
6. **Update**
7. **Ignore**
8. **Check-in**

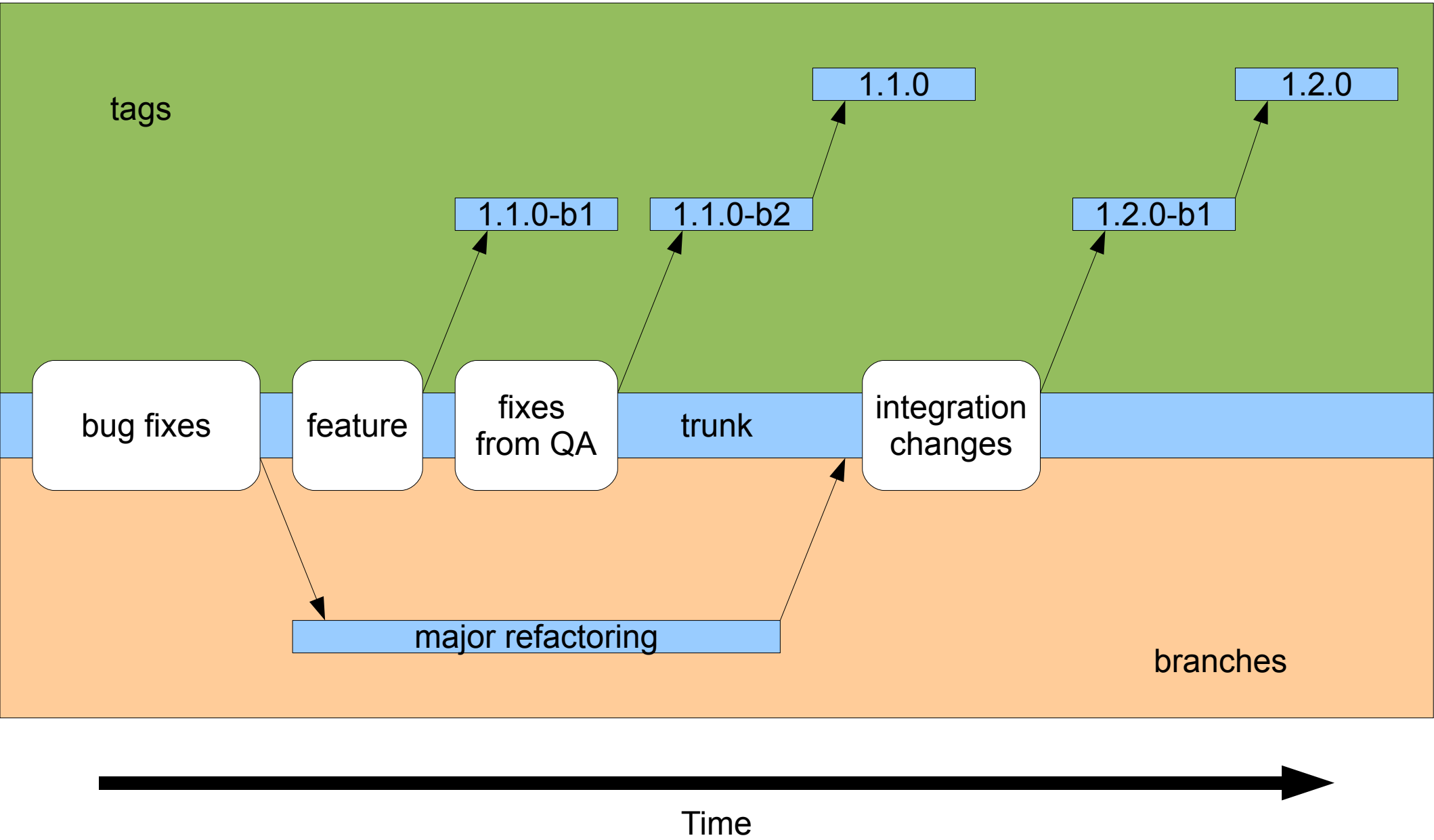
# optimistic locking

- Benefits
  - Concurrent development
  - Unobstructed use by CI server
  - Project management friendly

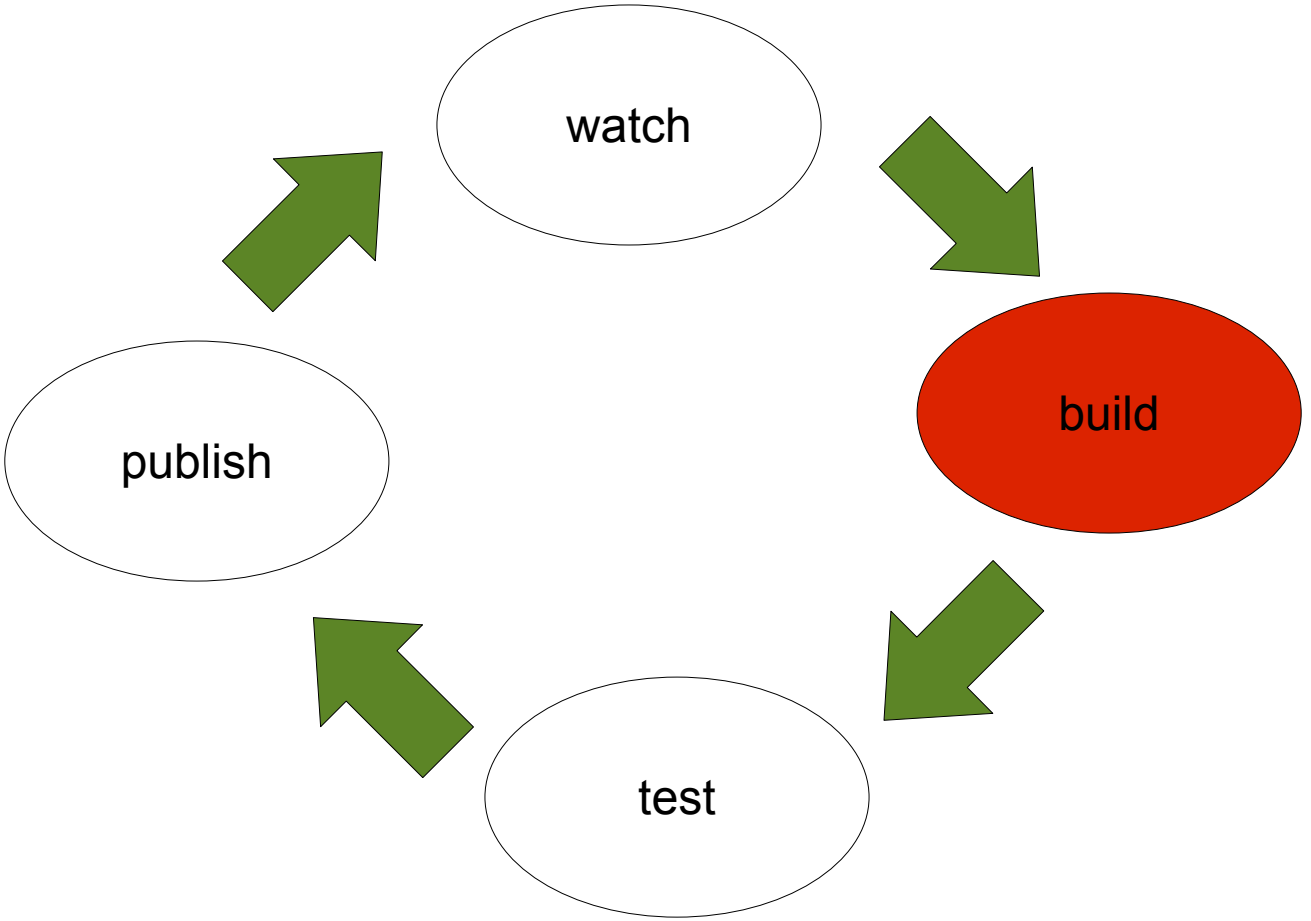
- Challenges
  - Resolving code conflicts can get complex
  - Potentially difficult learning curve

# folder management

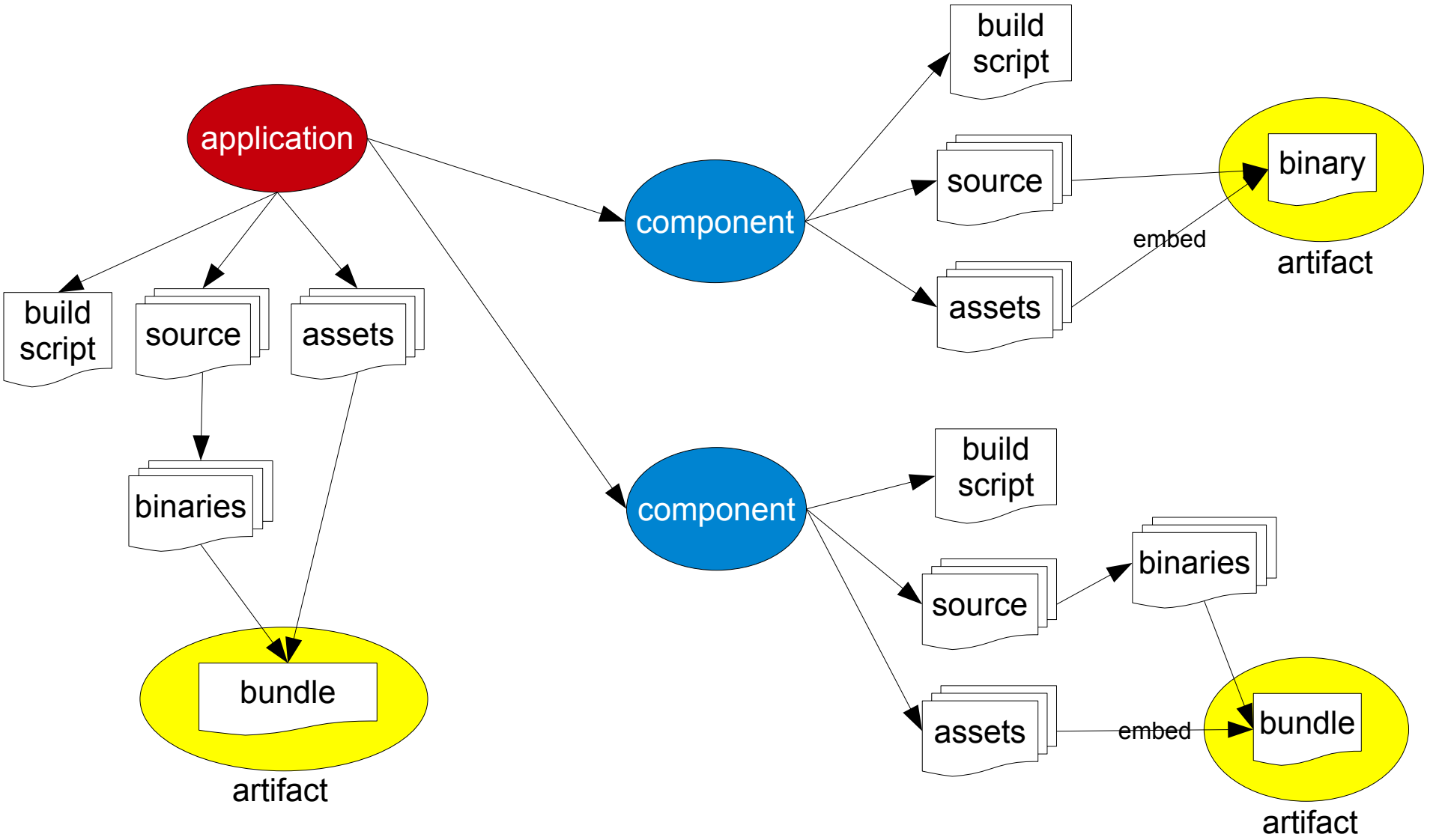
# example


tags
1.1.0
1.2.0
1.1.0-b1
1.1.0-b2
1.2.0-b1
bug fixes
feature
fixes
from QA
trunk
integration
changes
major refactoring
branches
Time

# the continuous integration loop

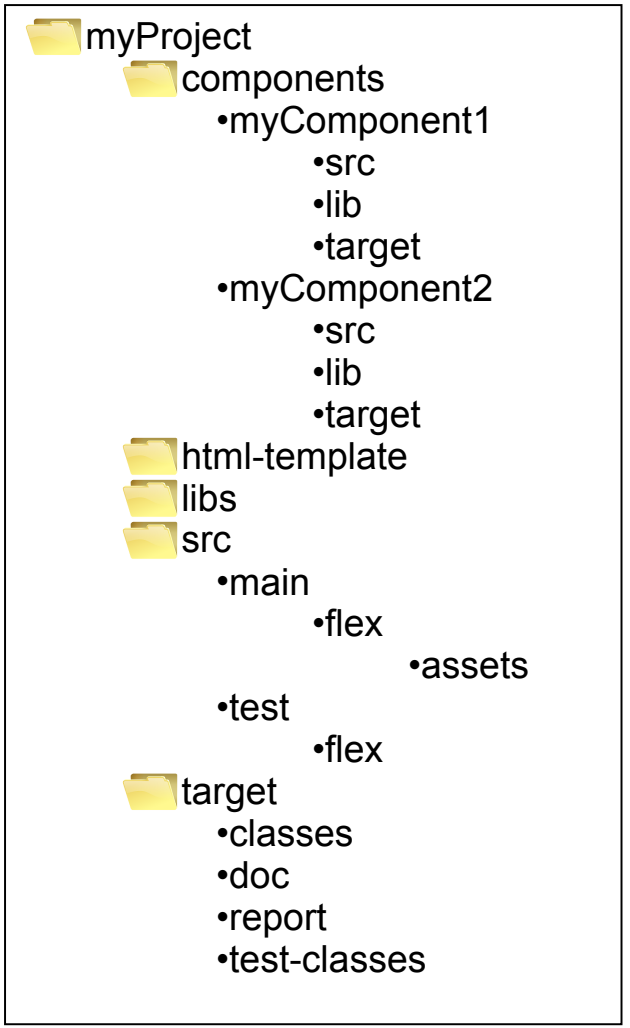


# build

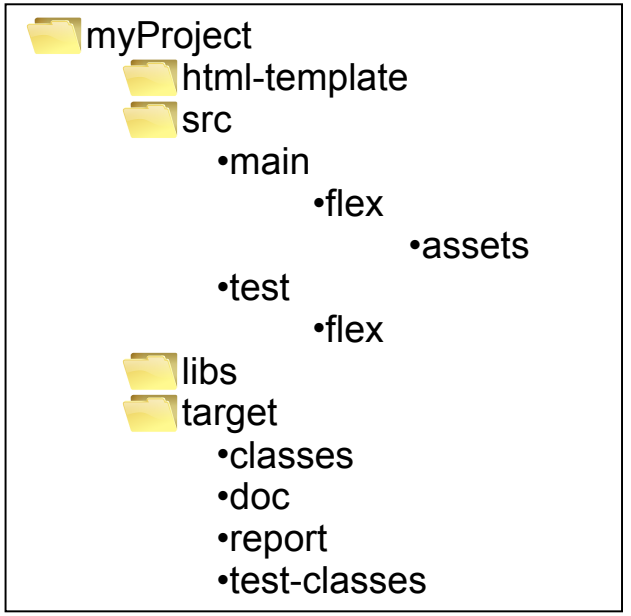

application
build script
source
assets
binaries
bundle
artifact
component
build script
source
assets
binary
embed
artifact
component
build script
source
binaries
assets
embed
bundle
artifact

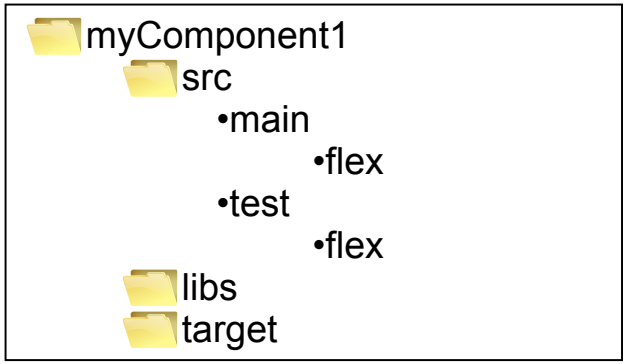
# project organization

## Nested




## Individual



# build lifecycle

1. clean
2. init
3. build
4. test
5. package

# tools


supports

- flex sdk
- flexunit
- fluint
- html-wrapper
- asdoc

Built by:

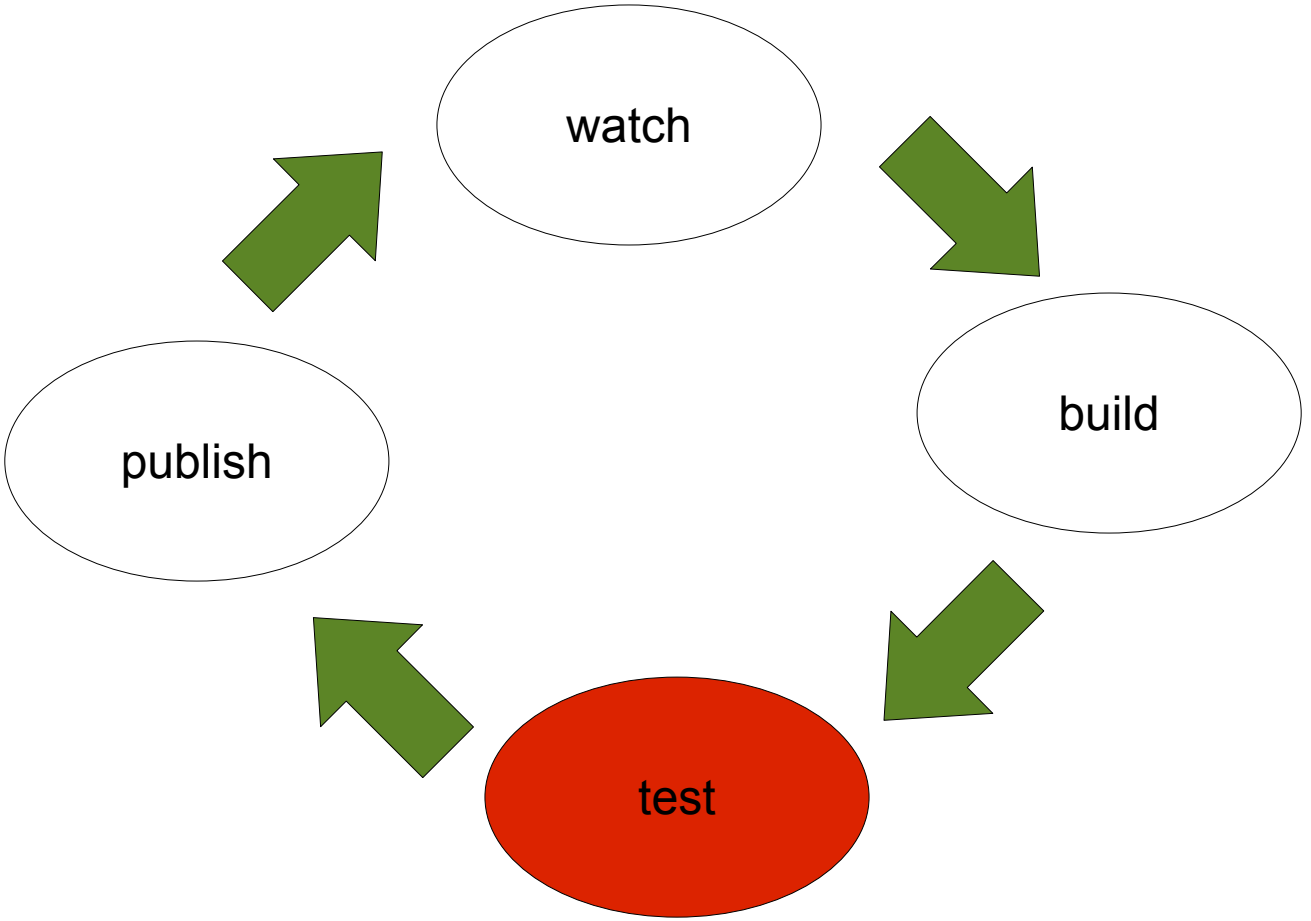
maven

supports

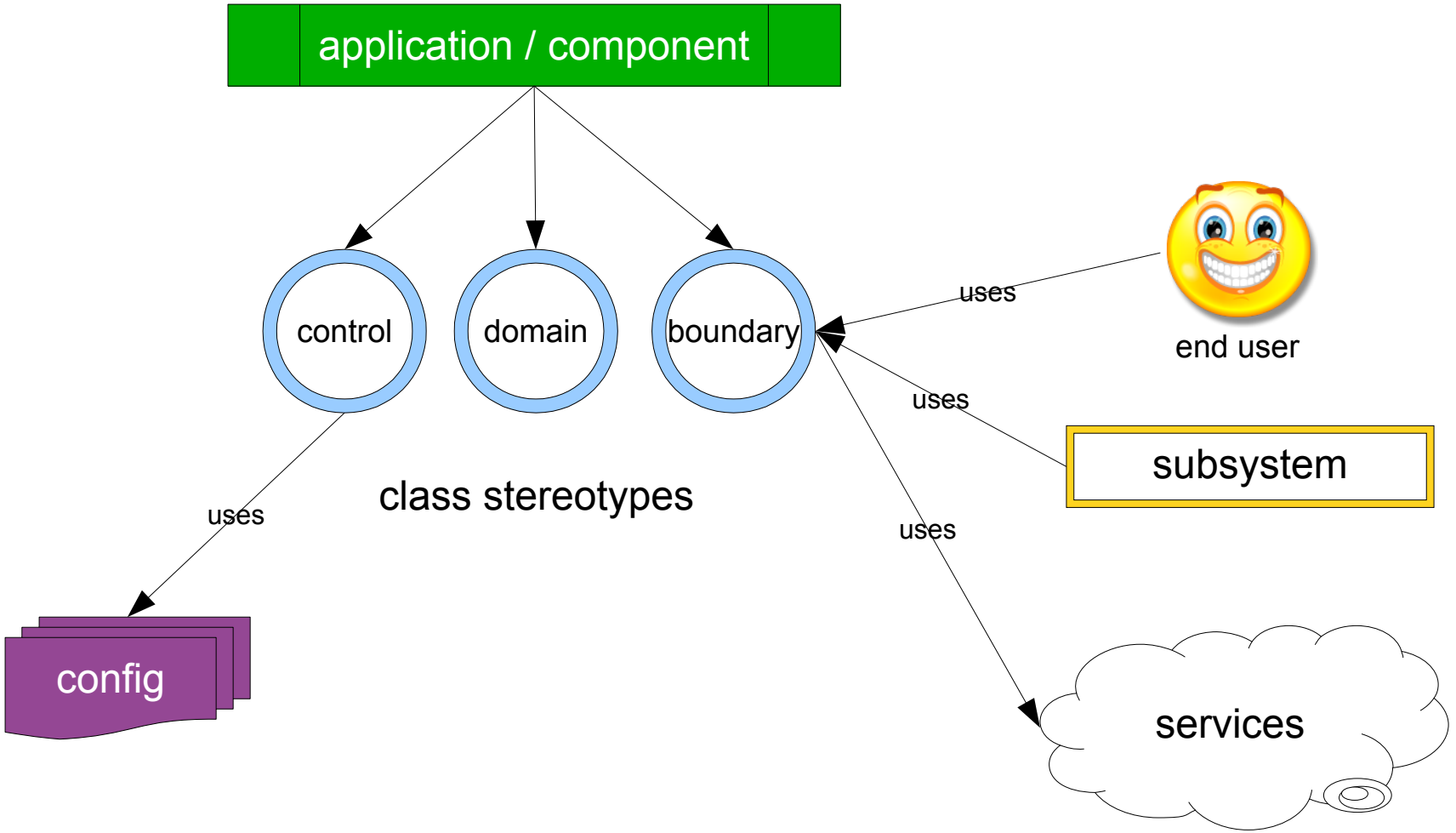
- flex-mojos
- flexunit
- fluint **
- html-wrapper
- asdoc

** - with a little help from Ant

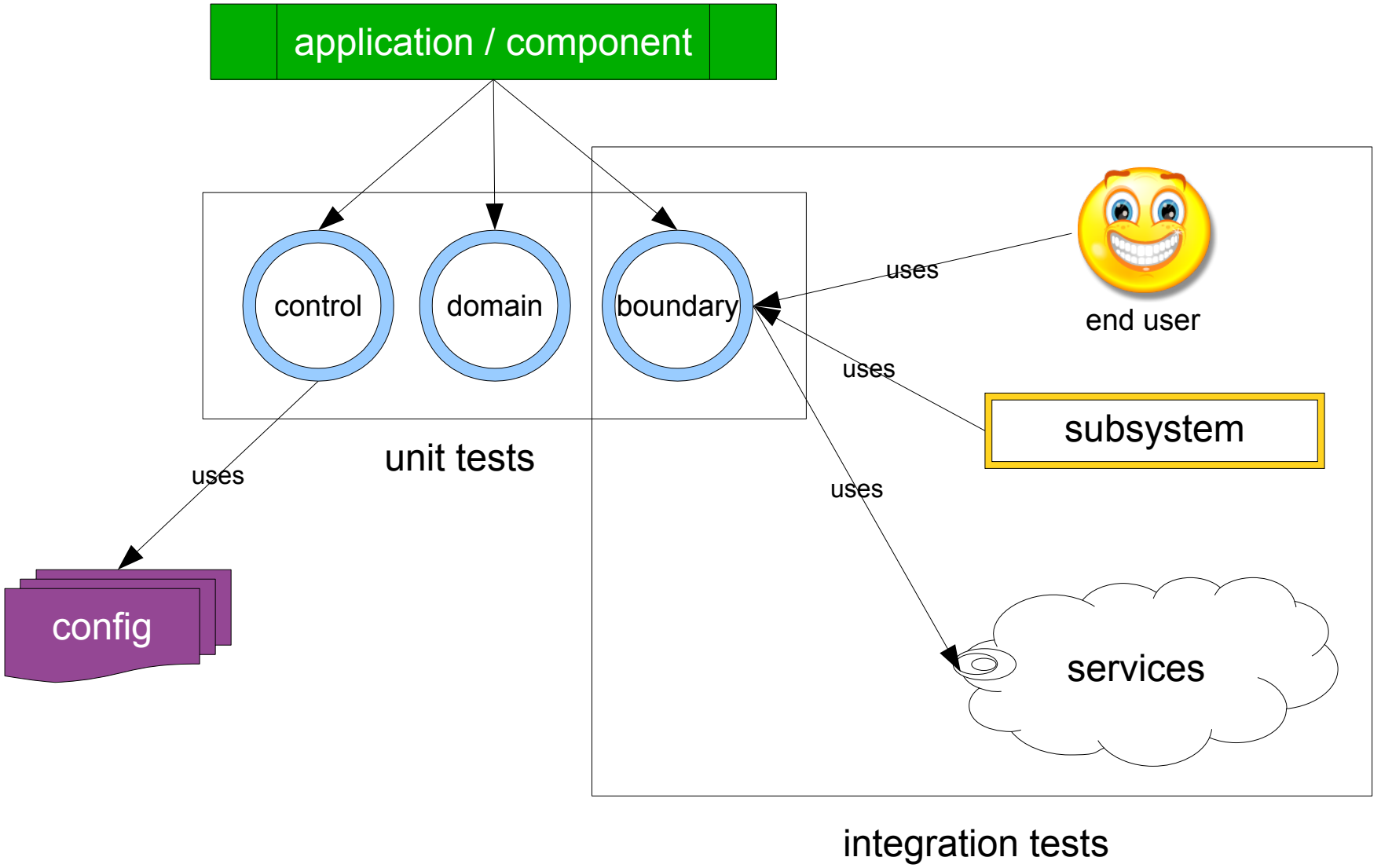
# the continuous integration loop



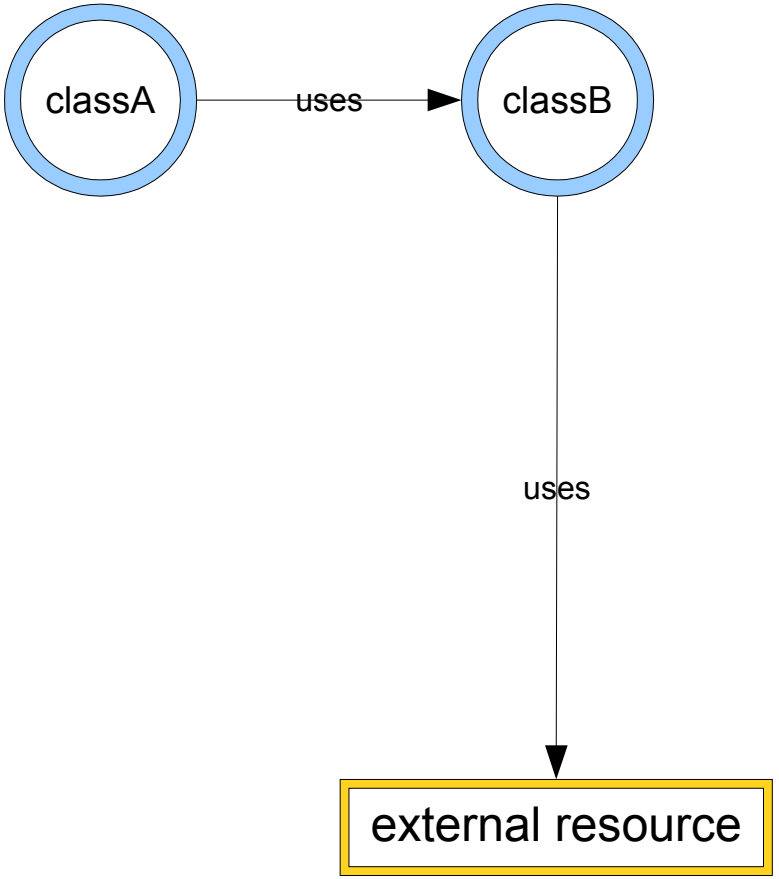
# test

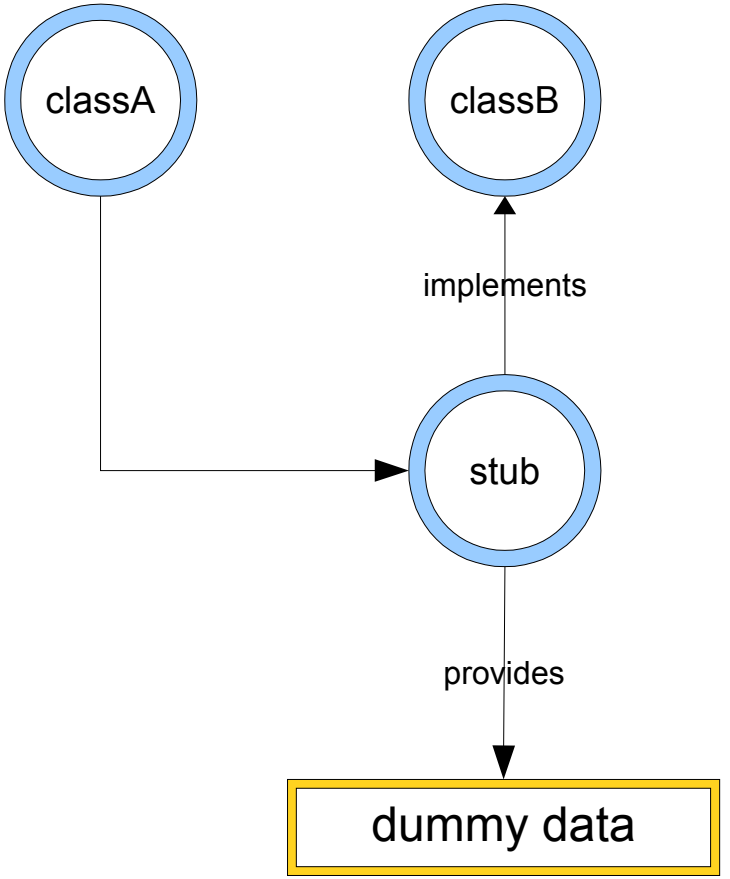

application / component
control
domain
boundary
uses
uses
uses
uses
end user
subsystem
class stereotypes
config
services

# test

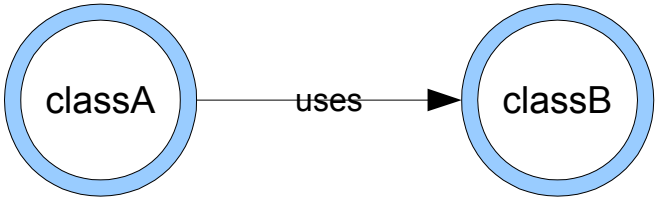

application / component
control
domain
boundary
unit tests
uses
config
uses
uses
end user
uses
subsystem
uses
services
integration tests

# stubs

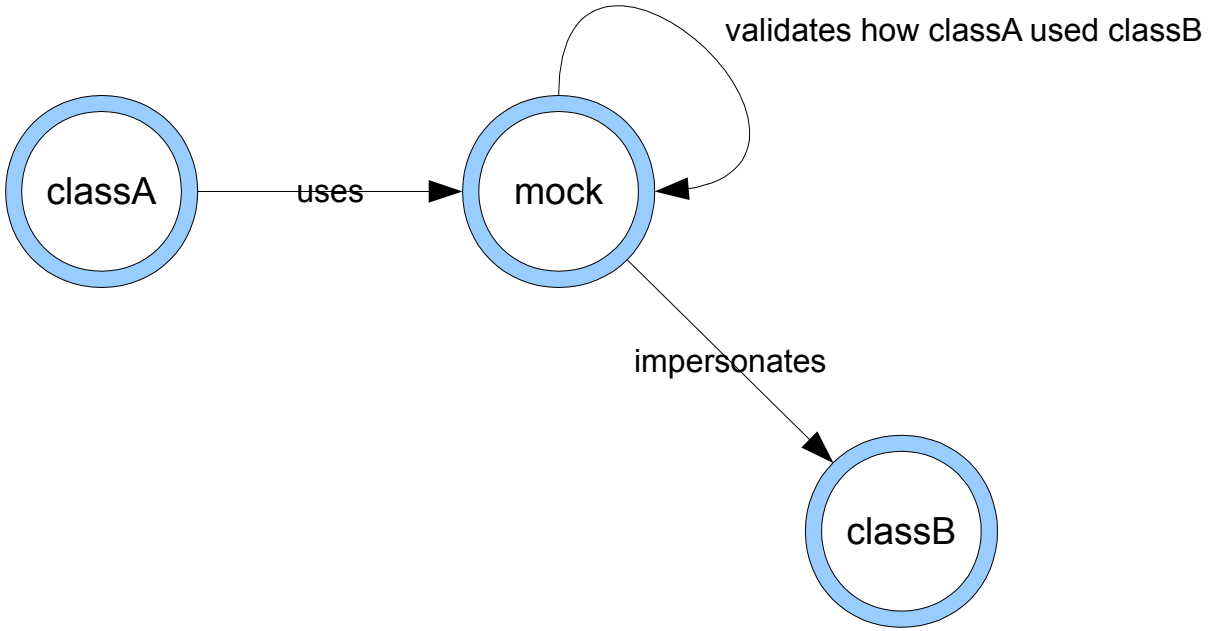

classA
uses
classB
uses
external resource


classA
classB
implements
stub
provides
dummy data

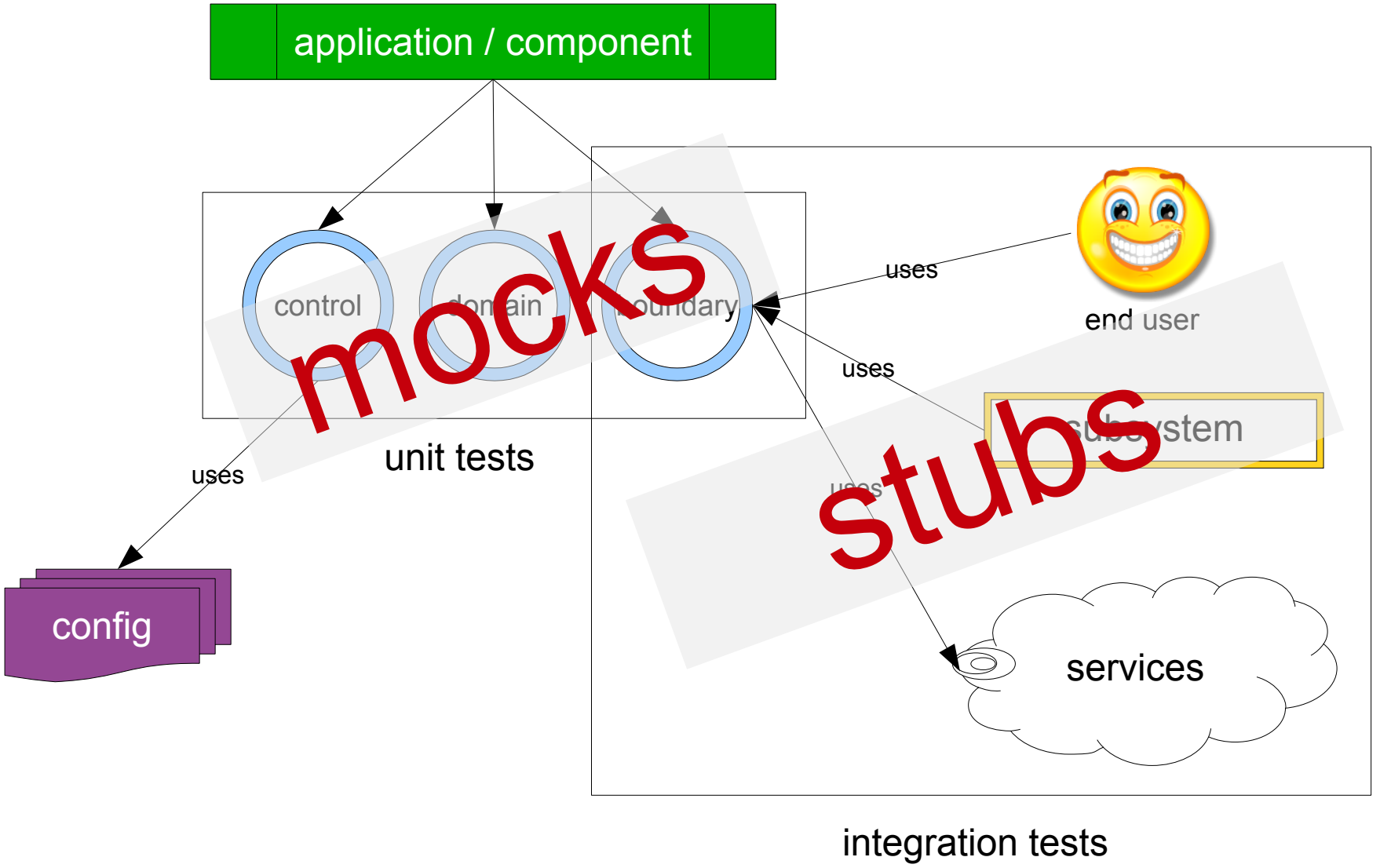
# mock objects



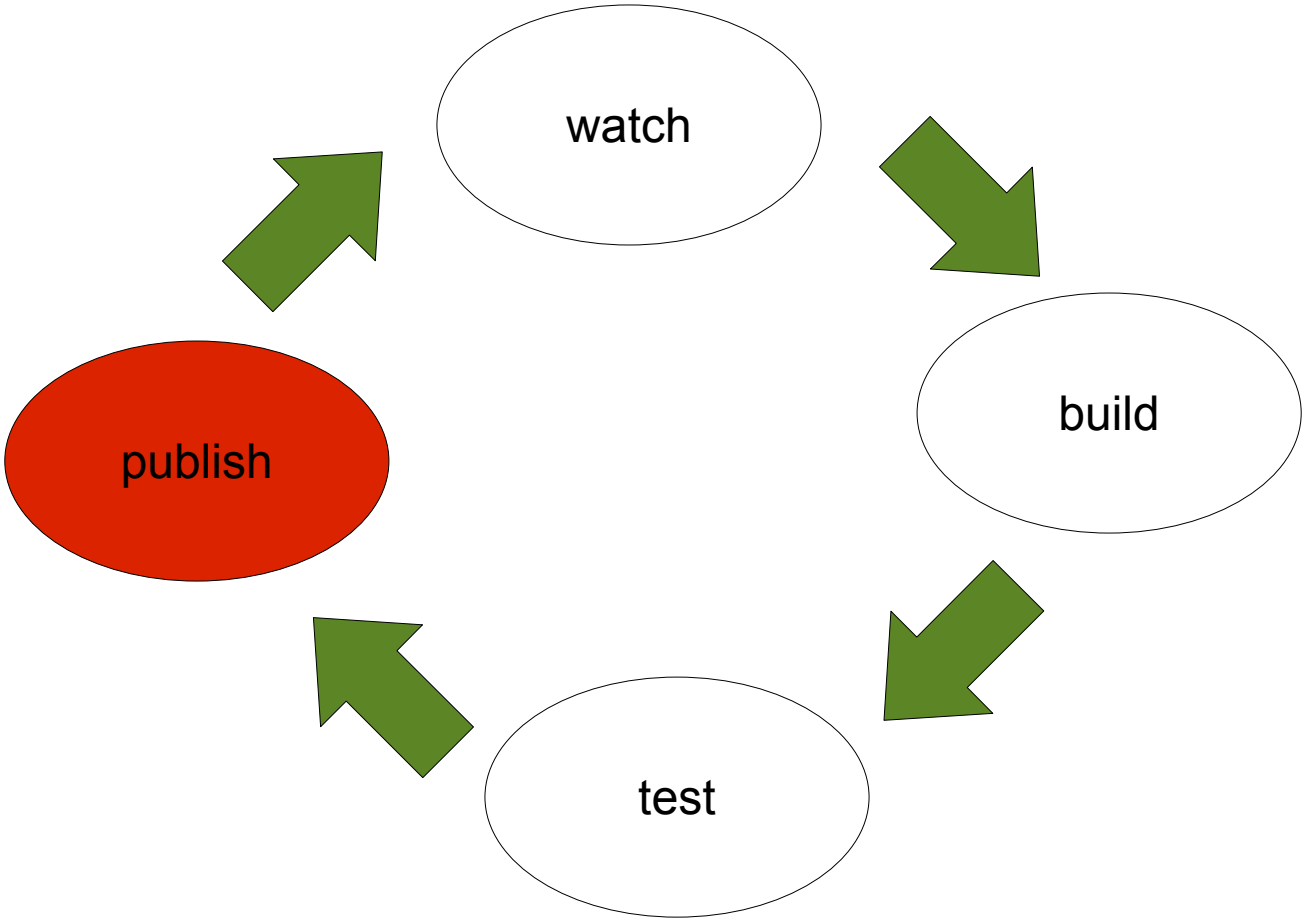

---



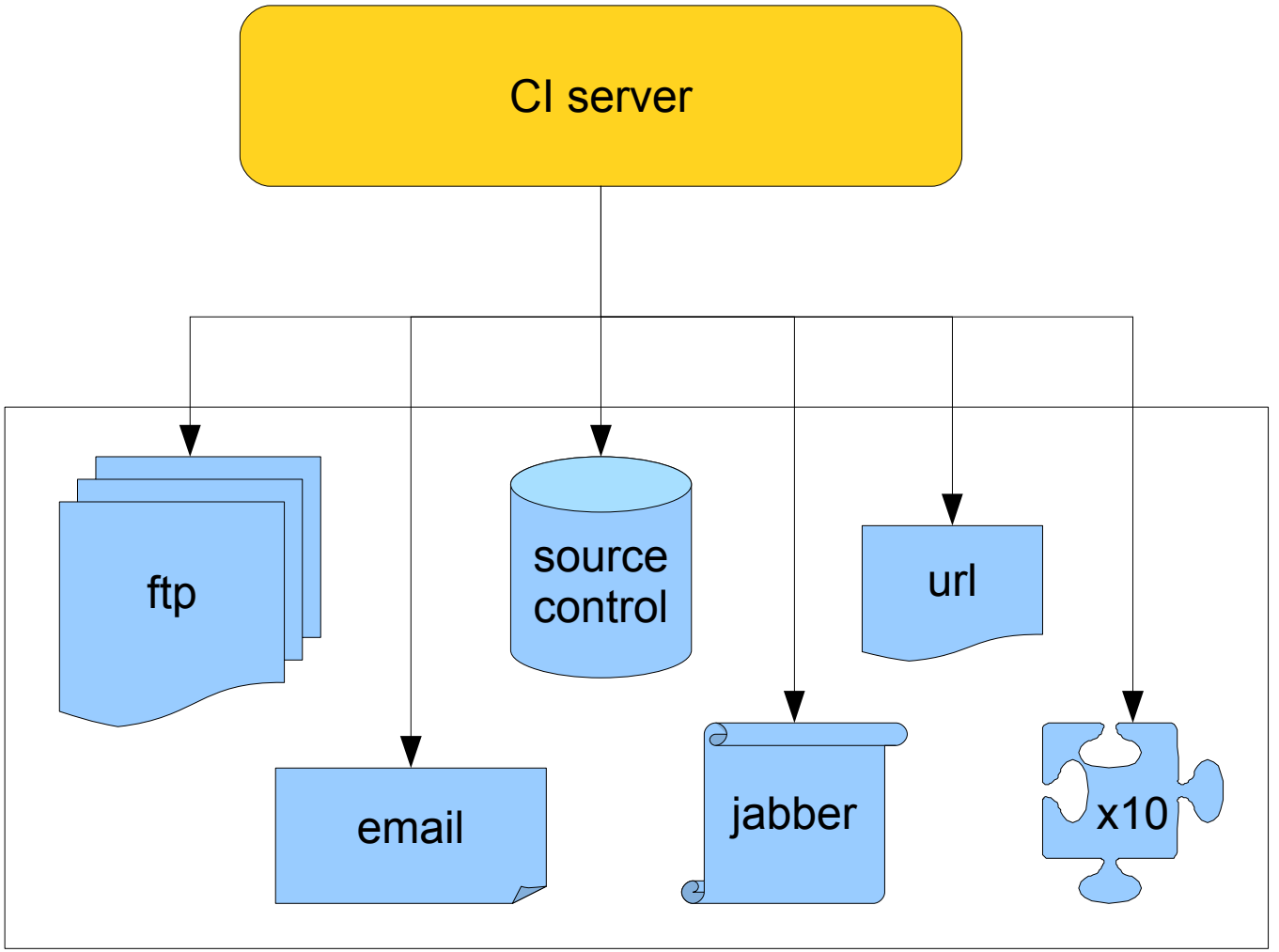
# automated regression



# the continuous integration loop



# publish



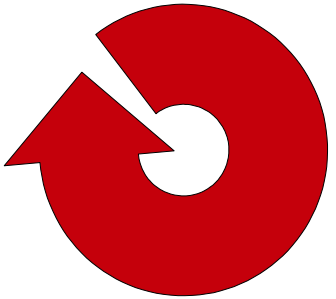
# outcome

success == build ran to completion, exited with no errors, and test reports show no failures

failure == build was halted, exited with errors, or test reports show failures

# feedback loop

broken build



fix, fix, fix

# how to get started

1. organize your project
2. write integration tests
3. create build scripts
4. setup your CI server
5. add more unit tests

wake up :)