Transforming Legacy Systems

An introduction to

The Mikado Method

Daniel Brolund @danielbrolund
Ola Ellnestam @ellnestam

agical

ARE YOU MOVING AS FAST AS YOU CAN?

Transforming Legacy Systems

An introduction to

The Mikado Method

Daniel Brolund @danielbrolund
Ola Ellnestam @ellnestam

agical

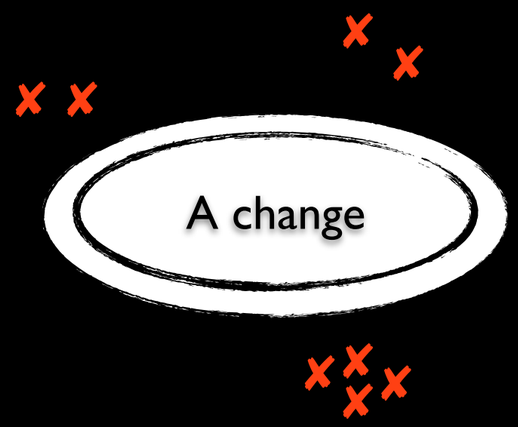
ARE YOU MOVING AS FAST AS YOU CAN?

The *Mikado* Method?

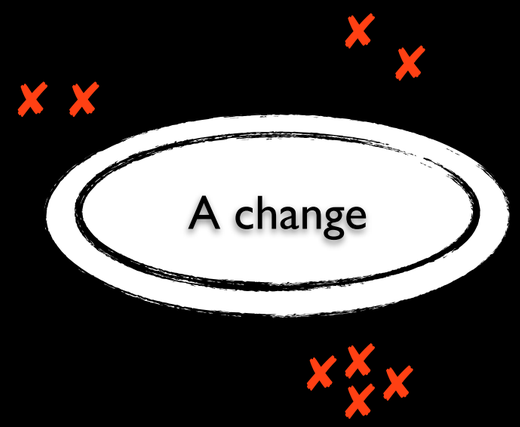
Changing a codebase...

Lets say this was our system...

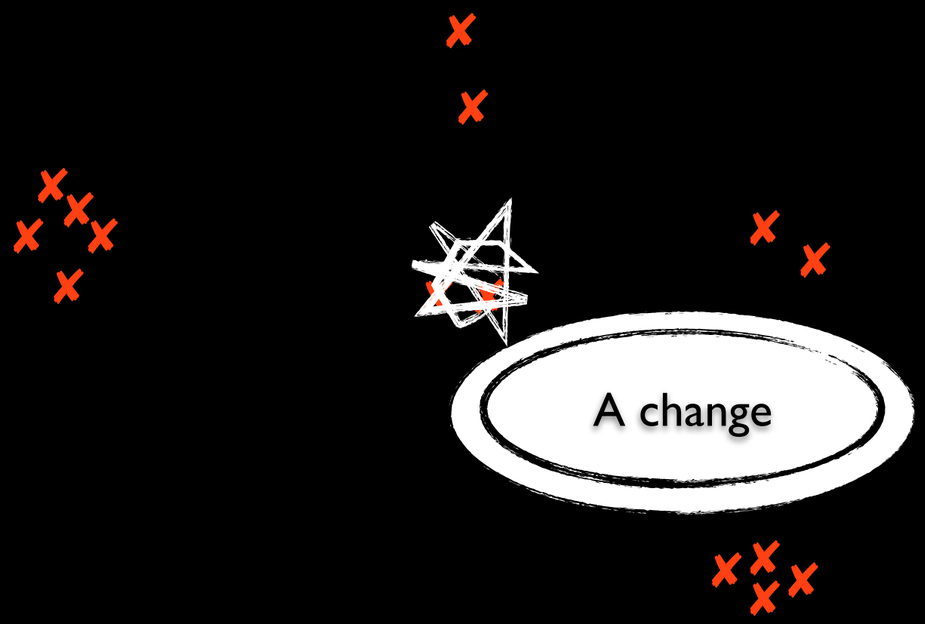
We were to make a change...



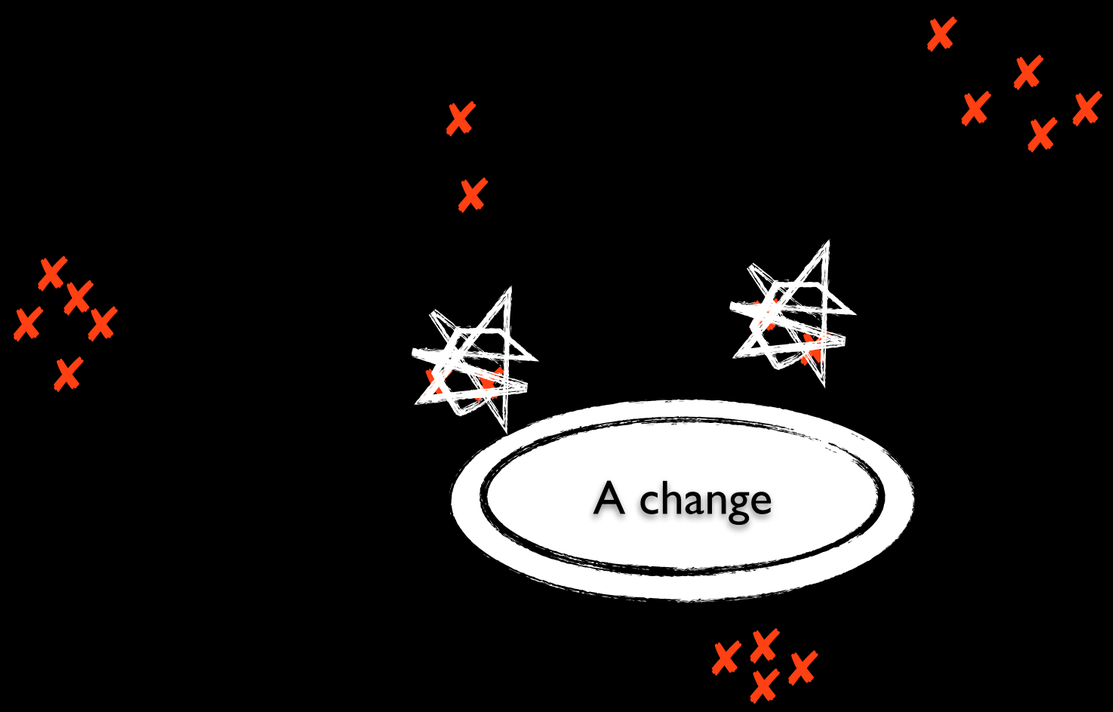
...but we got some errors...



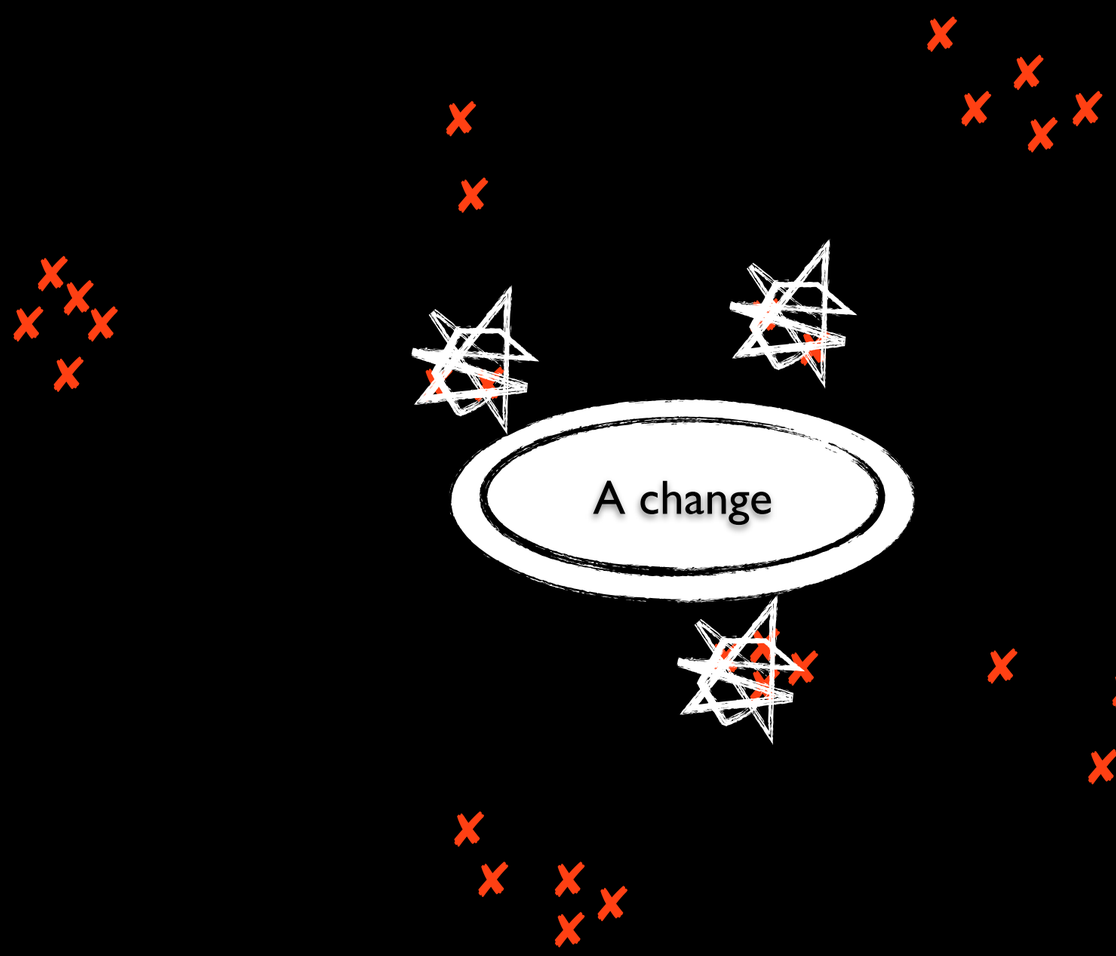
Patching led to more errors...



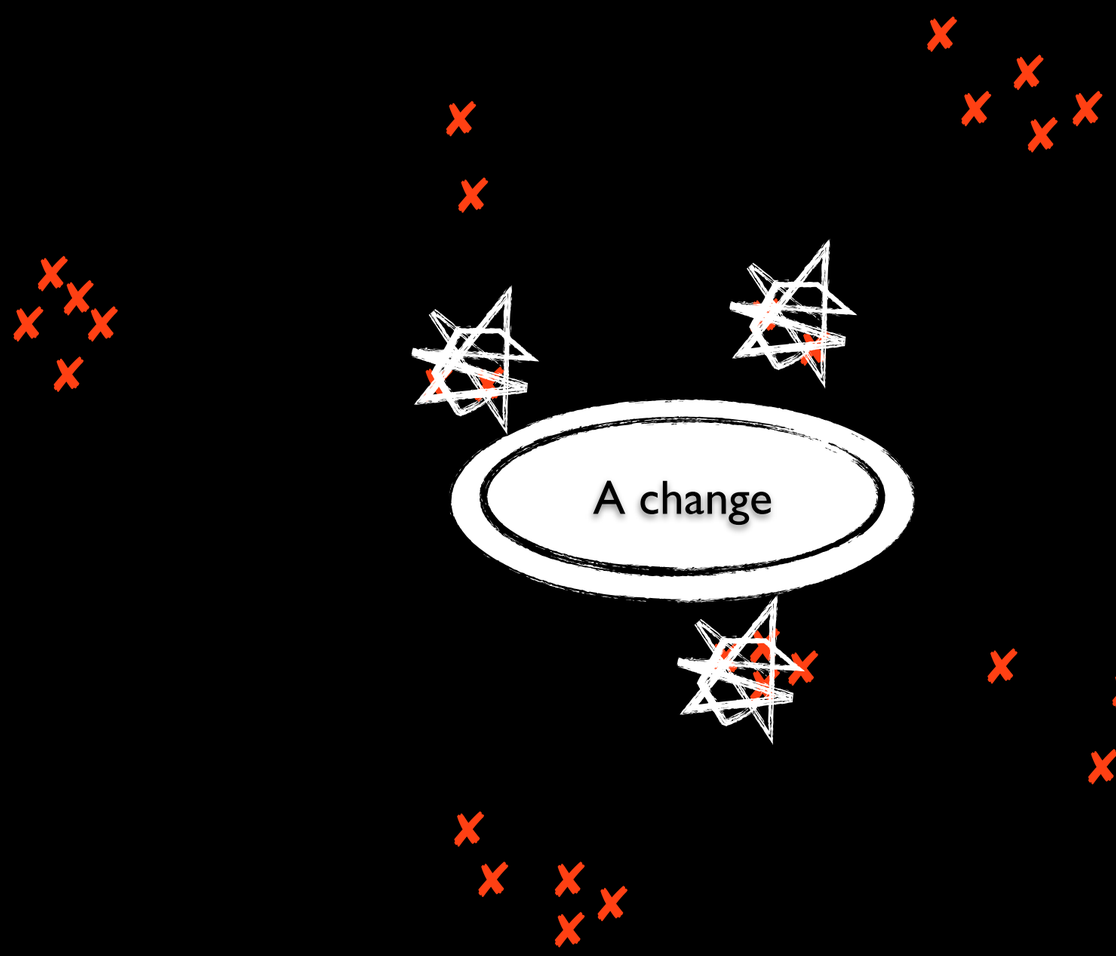
Patching led to more errors...



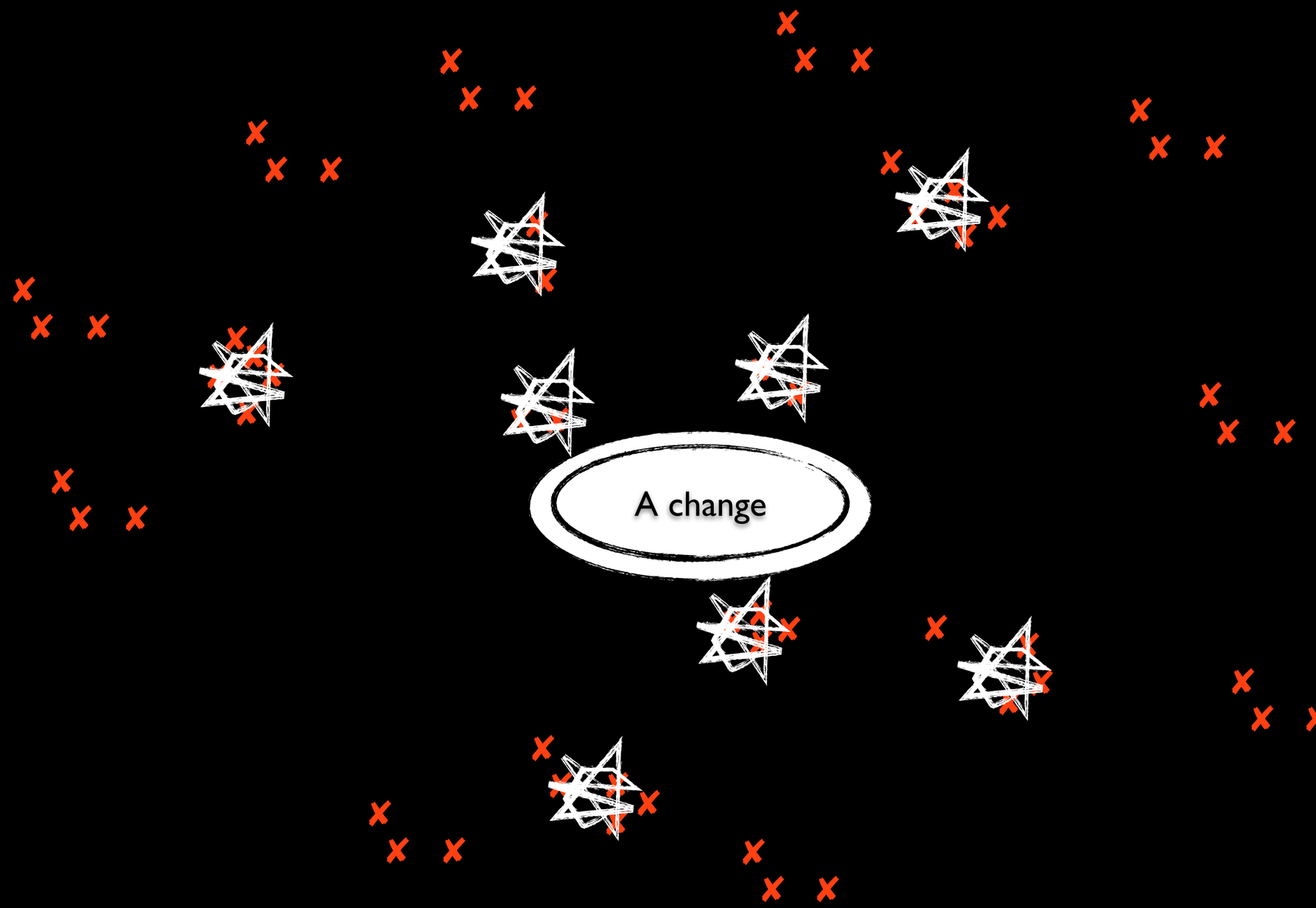
Patching led to more errors...



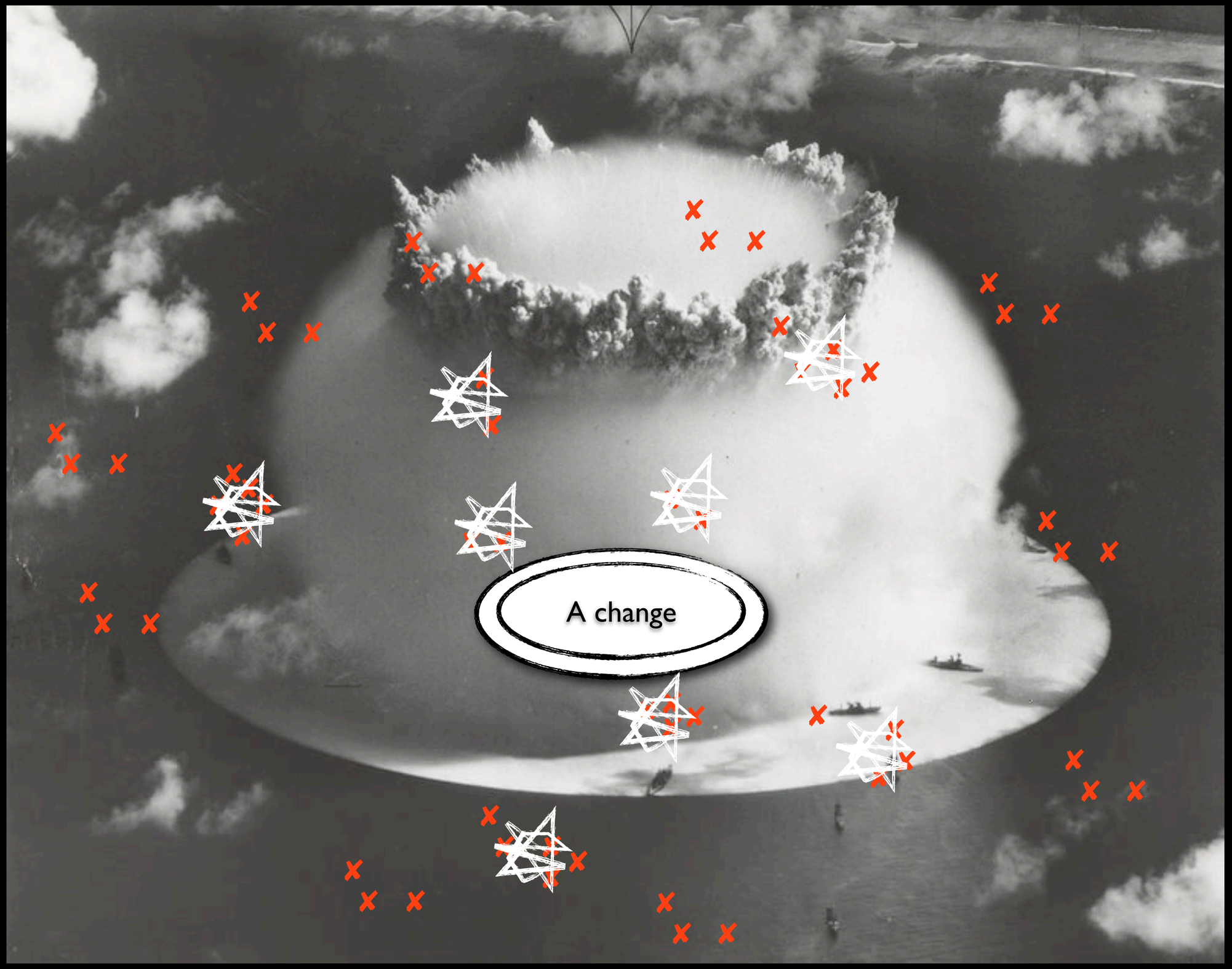
Patching led to more errors...



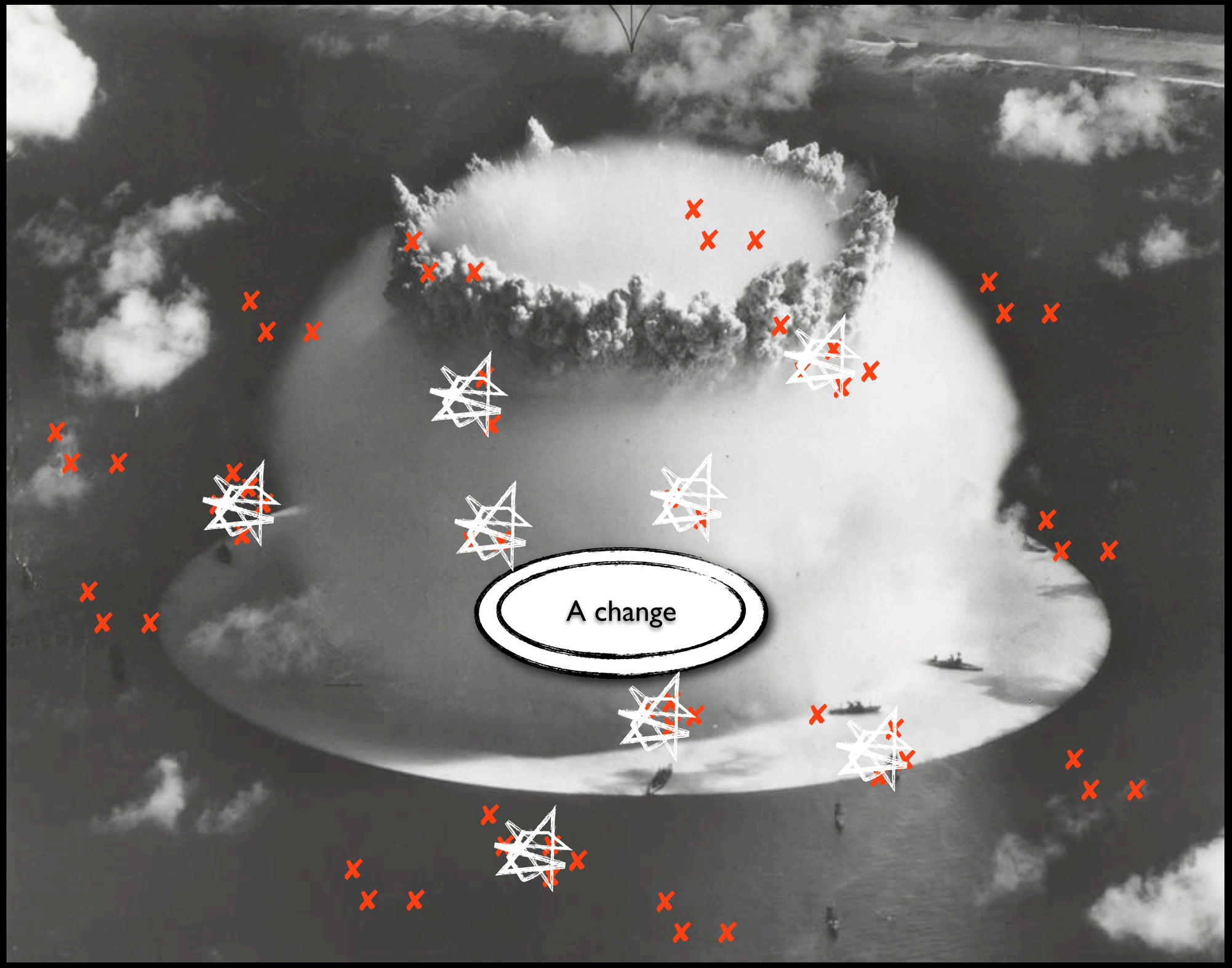
...and each patch led to even more errors...



...and each patch led to even more errors...



Like trying to stop the shockwave with our hands...



We could only revert.

We could only revert.

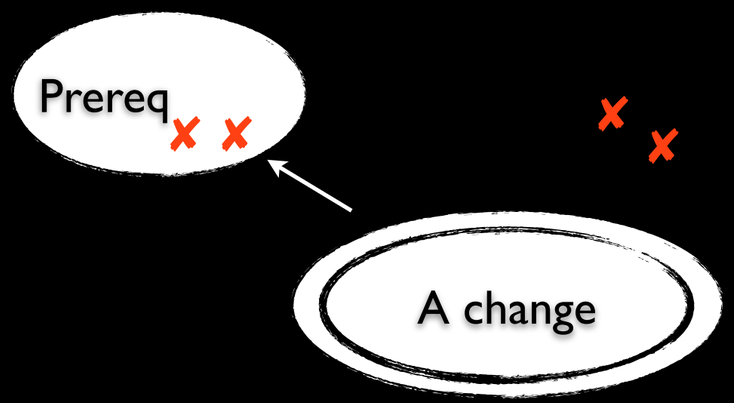
But we really
needed to do
that change!

The same
change...

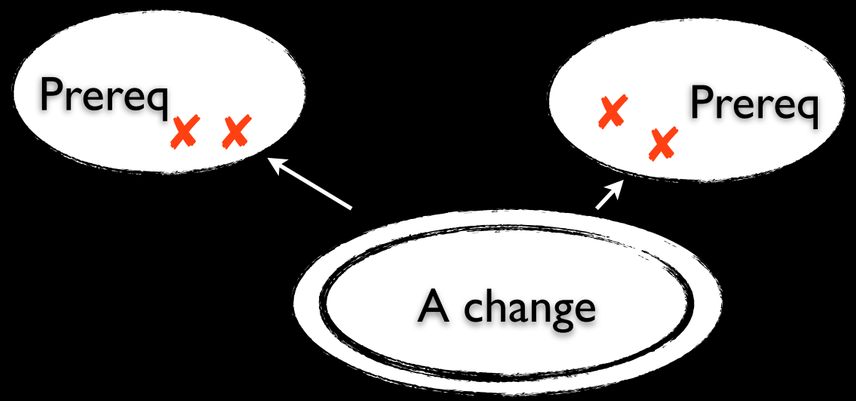
...the same
errors...

This time, we noted a prerequisite for each of the errors...

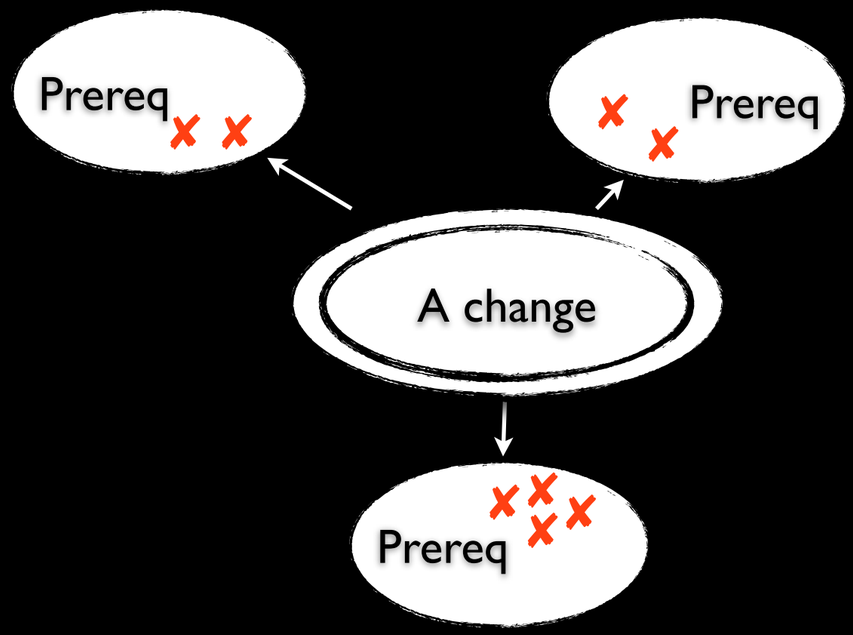
This time, we noted a prerequisite for each of the errors...



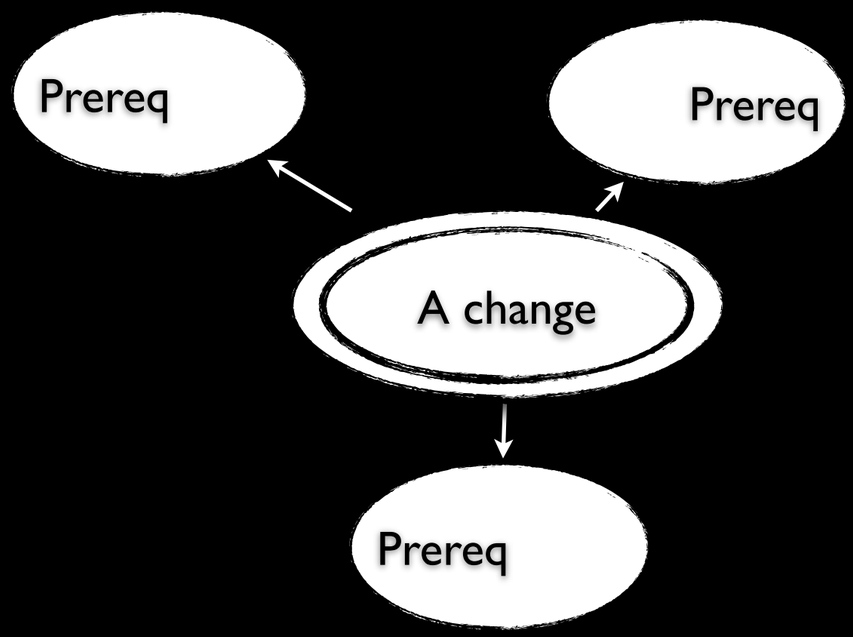
This time, we noted a prerequisite for each of the errors...



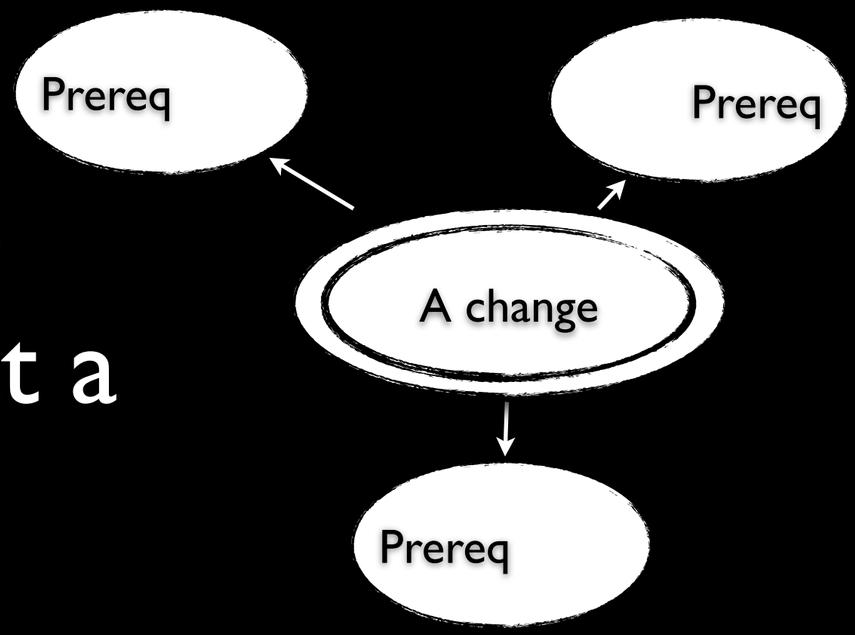
This time, we noted a prerequisite for each of the errors...



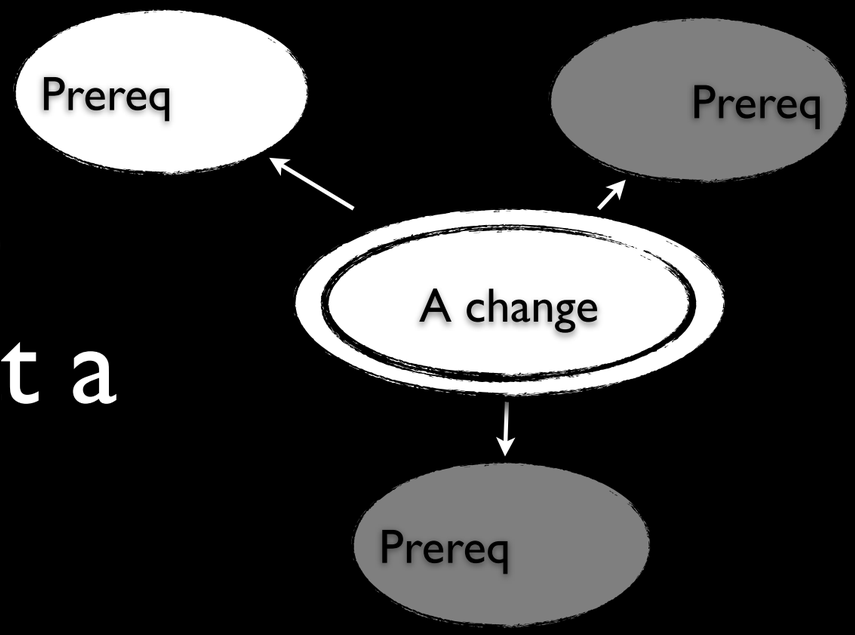
...then we reverted the errors,
but we kept the notes!



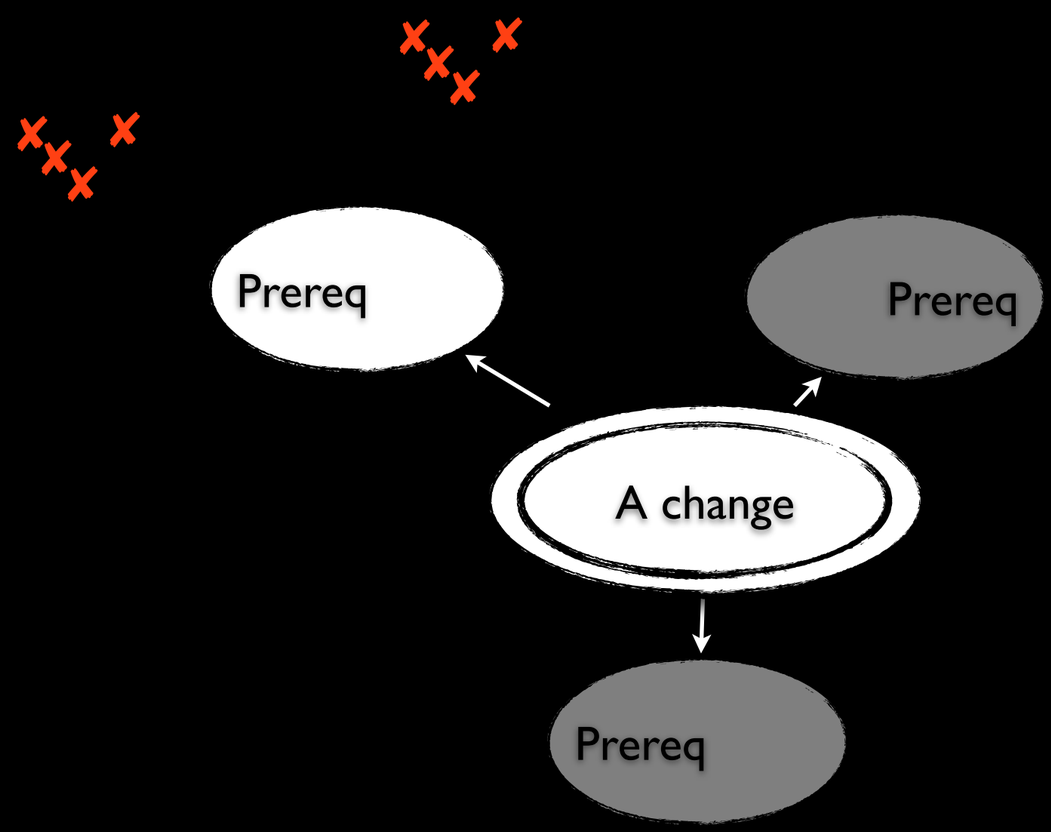
Then we implemented the prerequisites, one at a time...



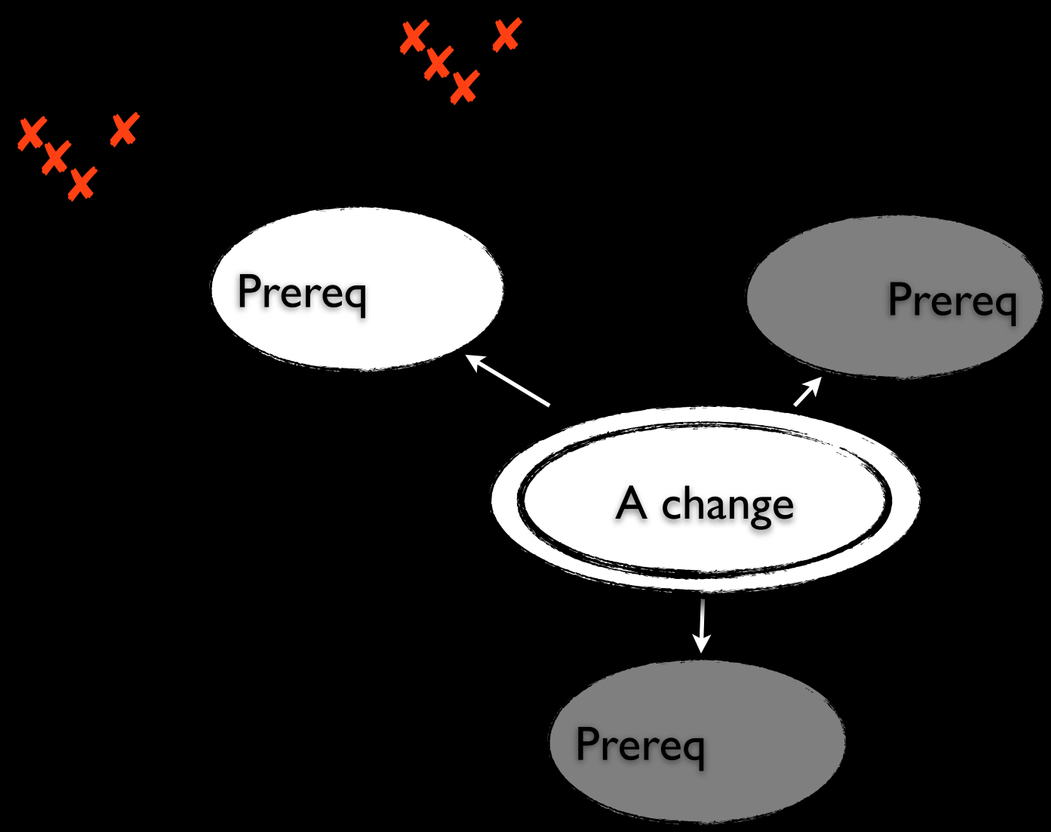
Then we implemented the prerequisites, one at a time...



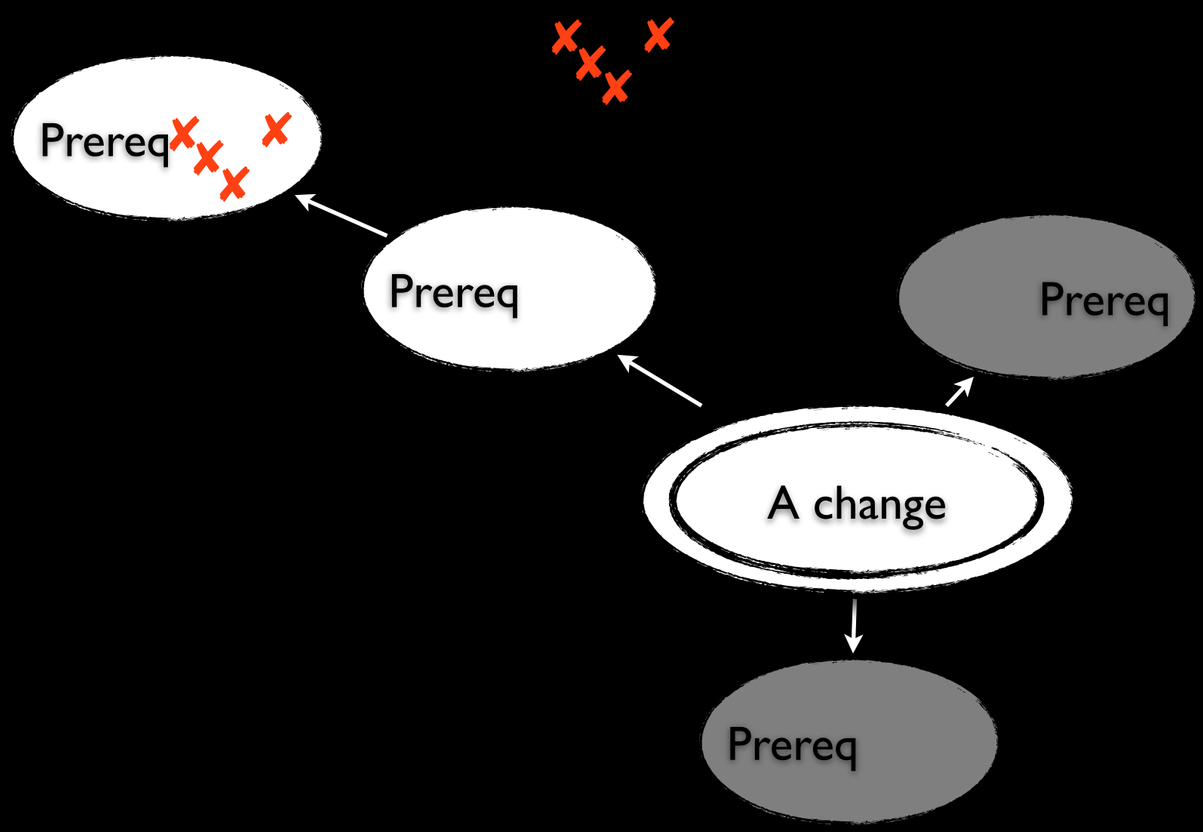
...got new errors...



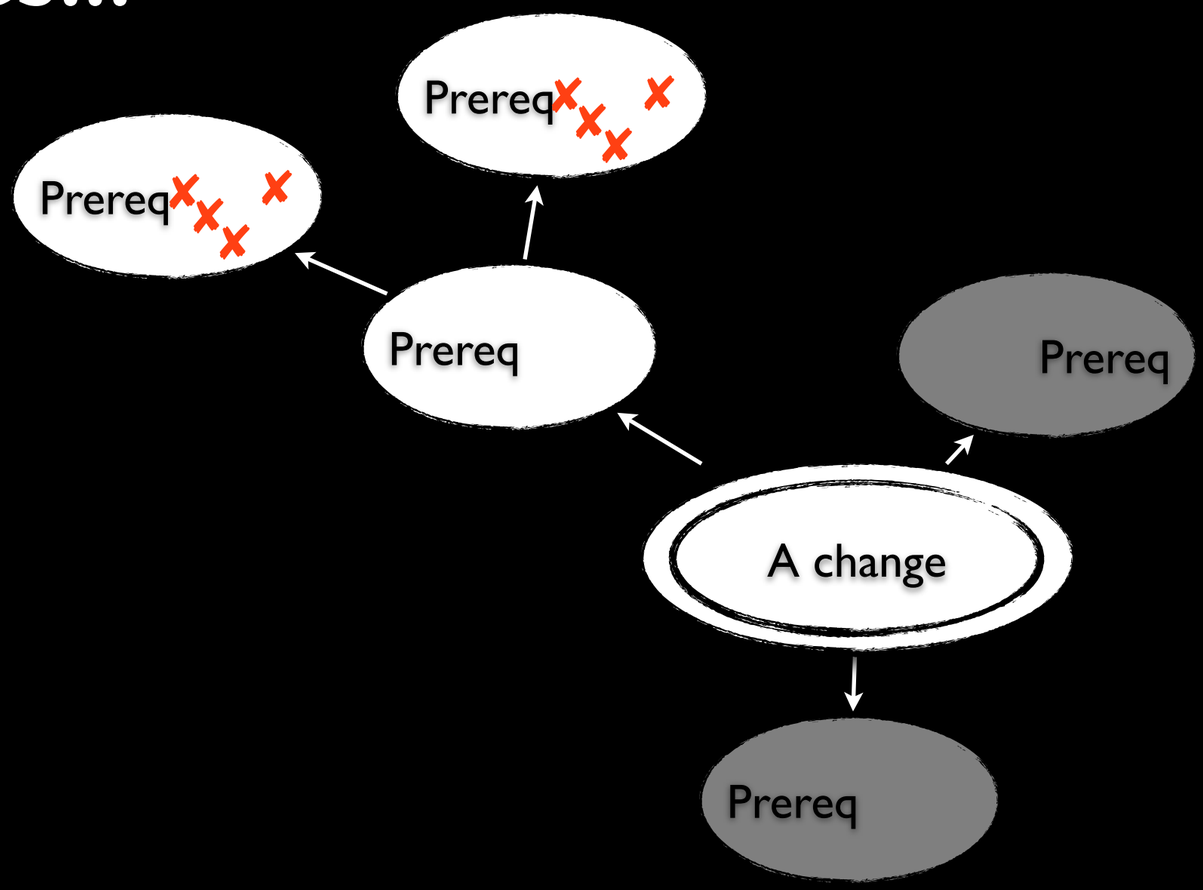
...noted the new prerequisites...



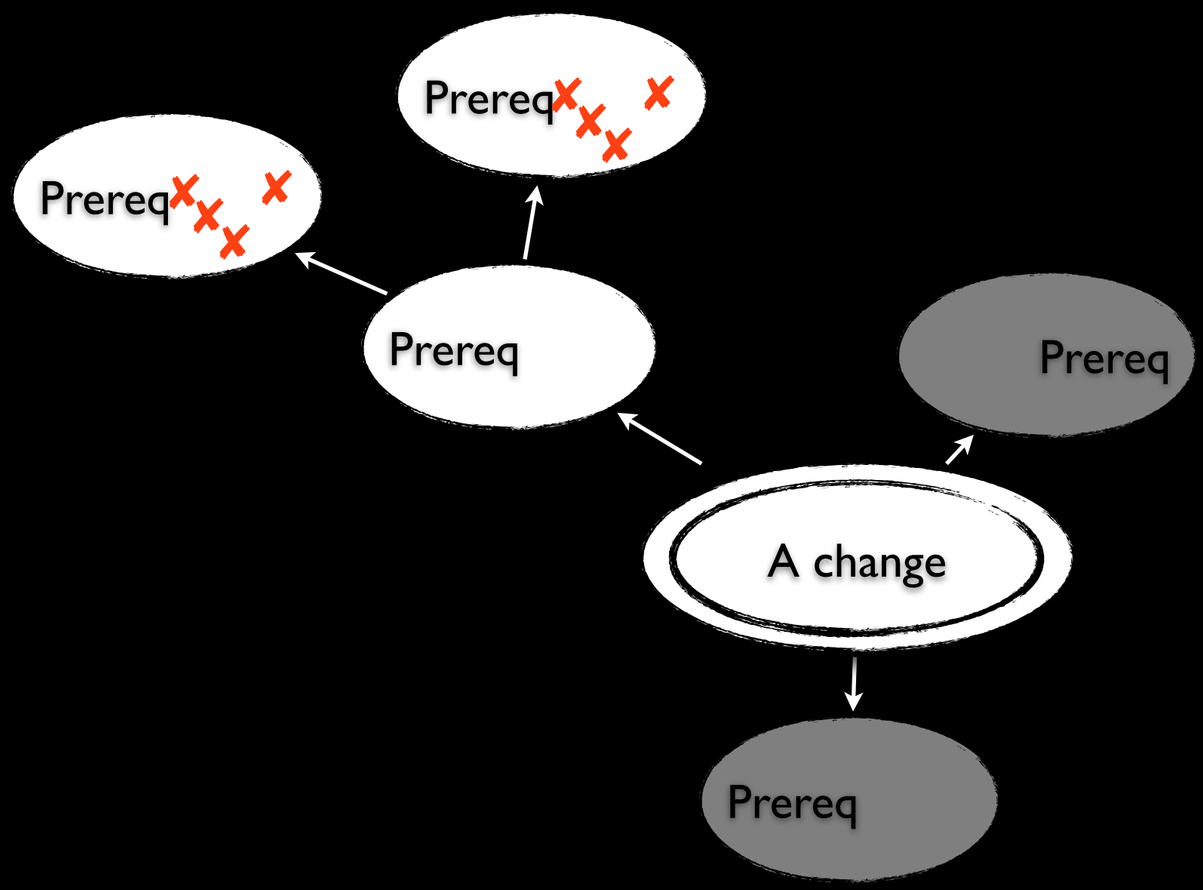
...noted the new prerequisites...



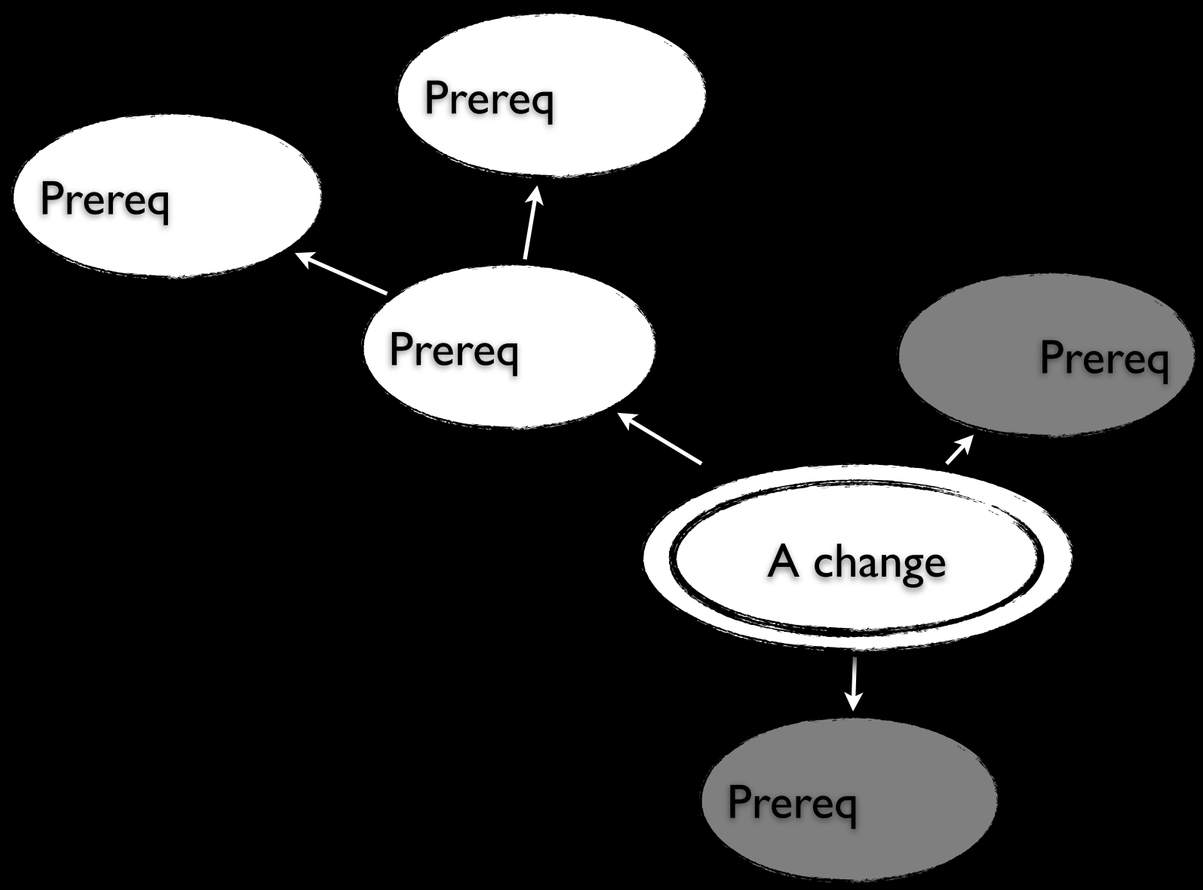
...noted the new prerequisites...



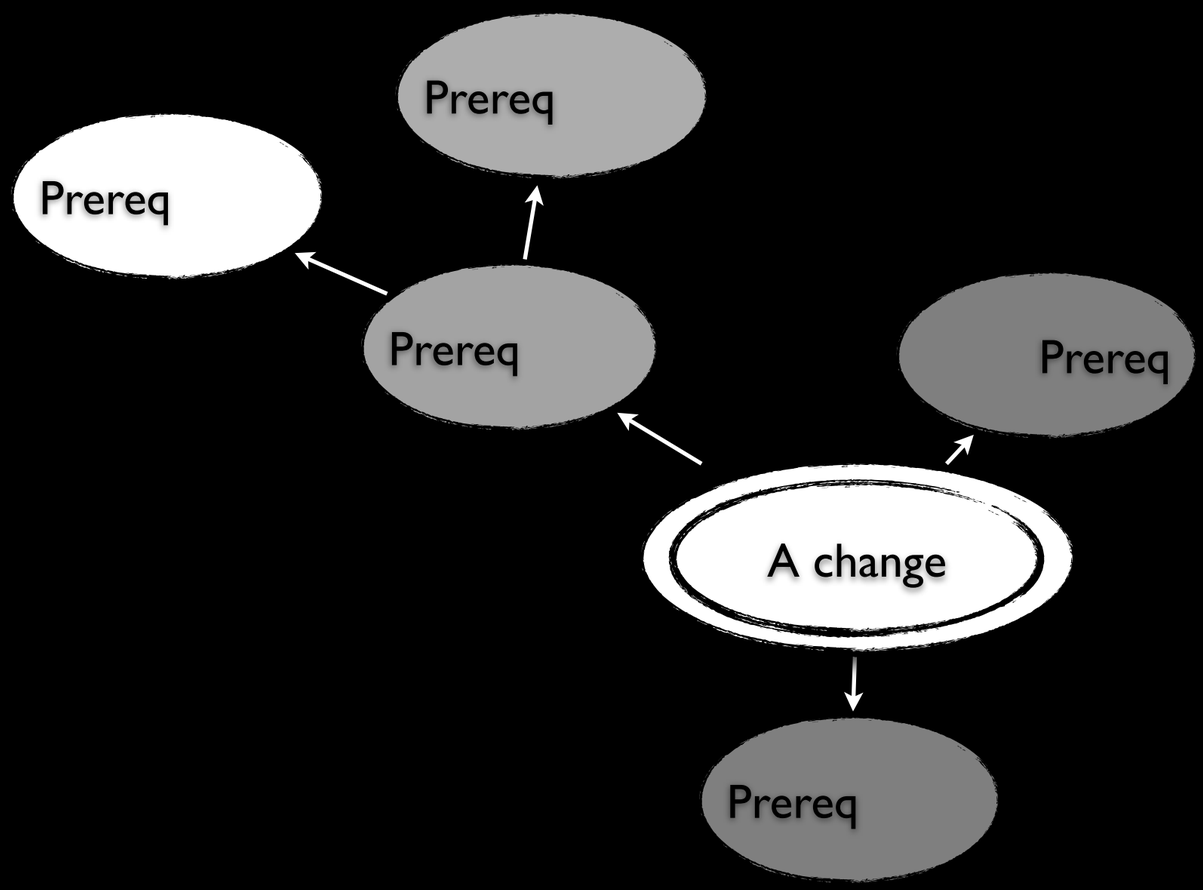
...and reverted again



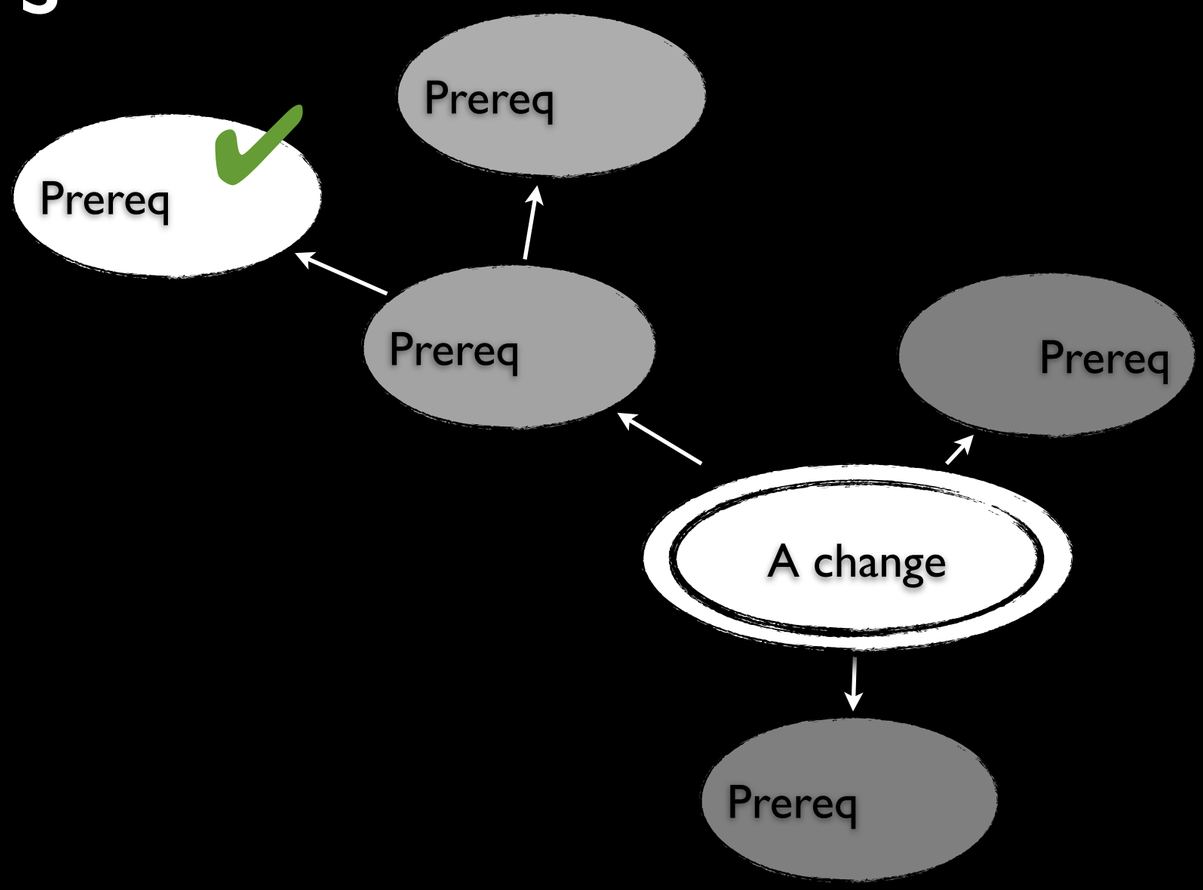
...and reverted again



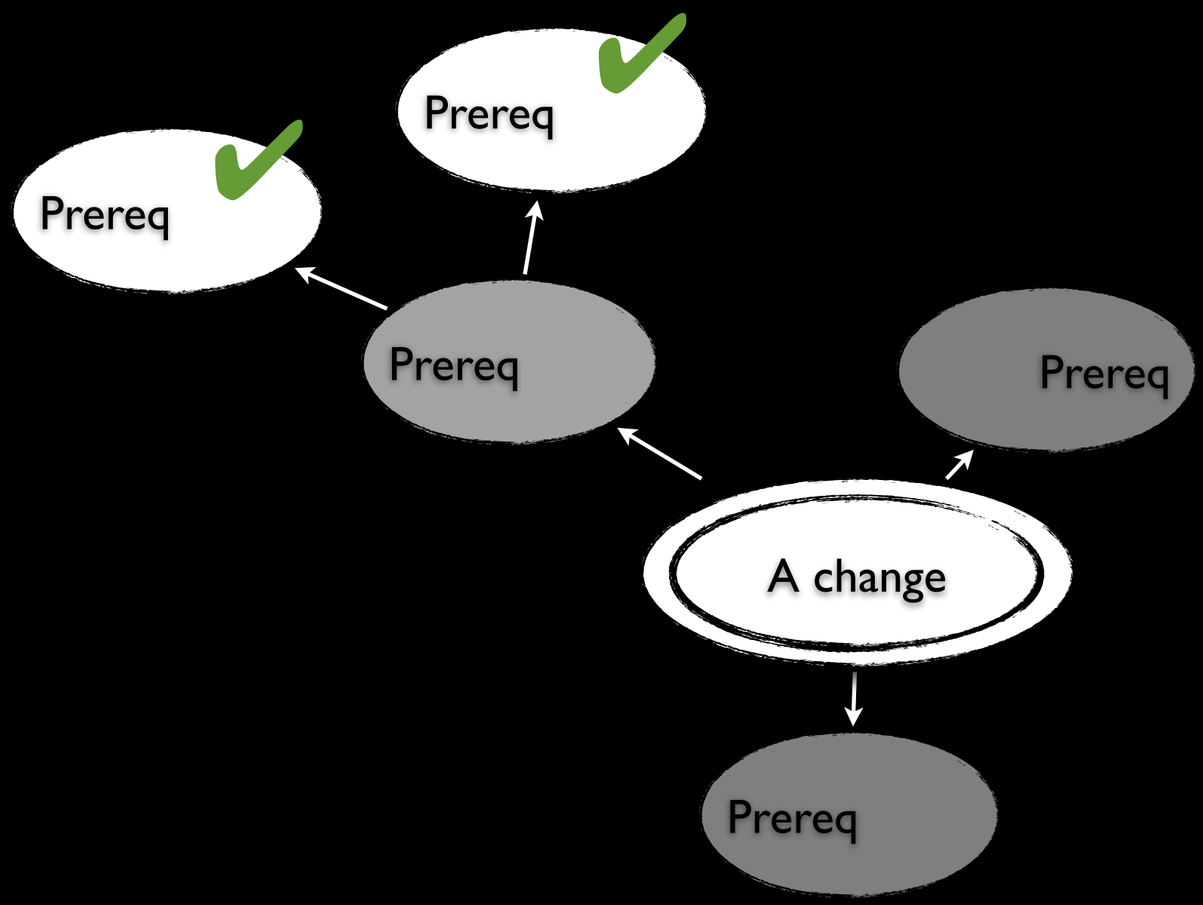
Picked the next leaf a.s.o...



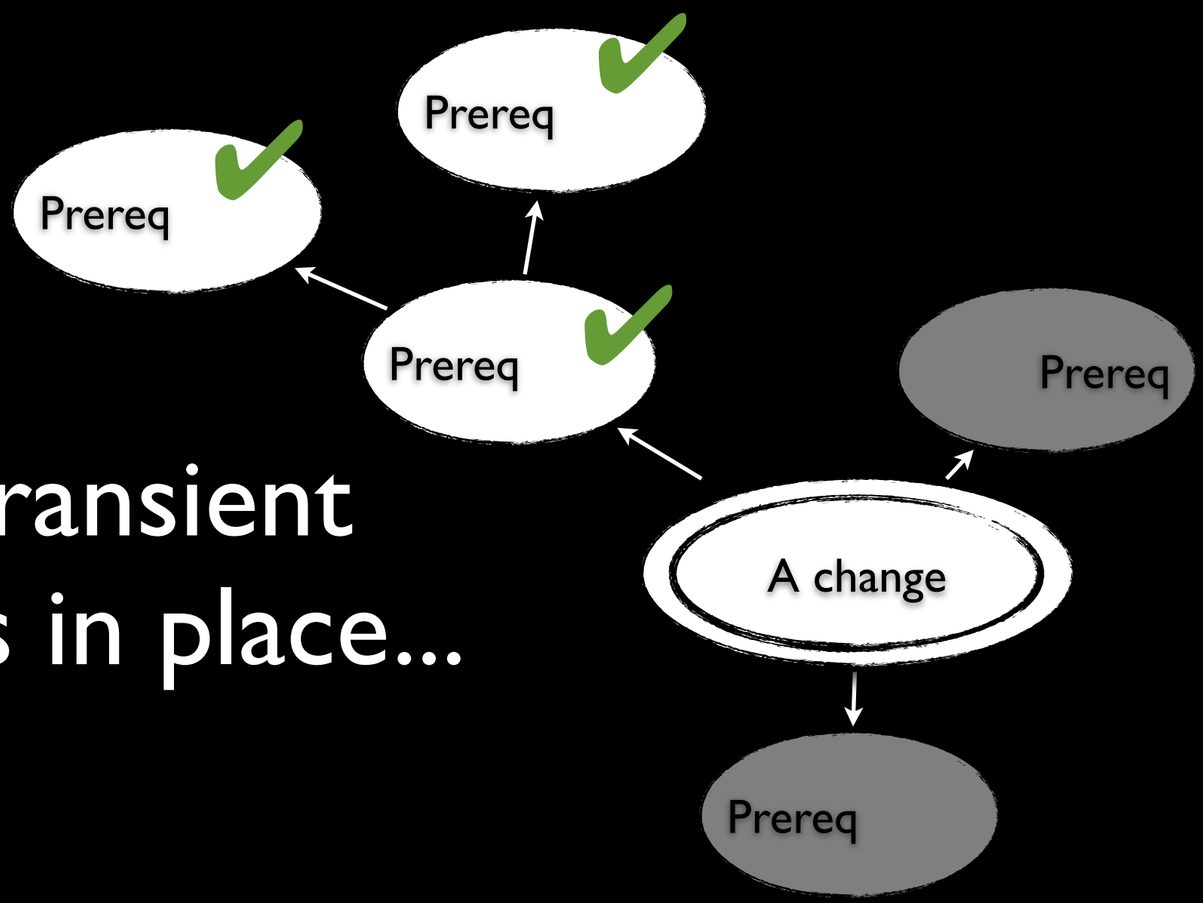
...until we could do a prerequisite w/o errors

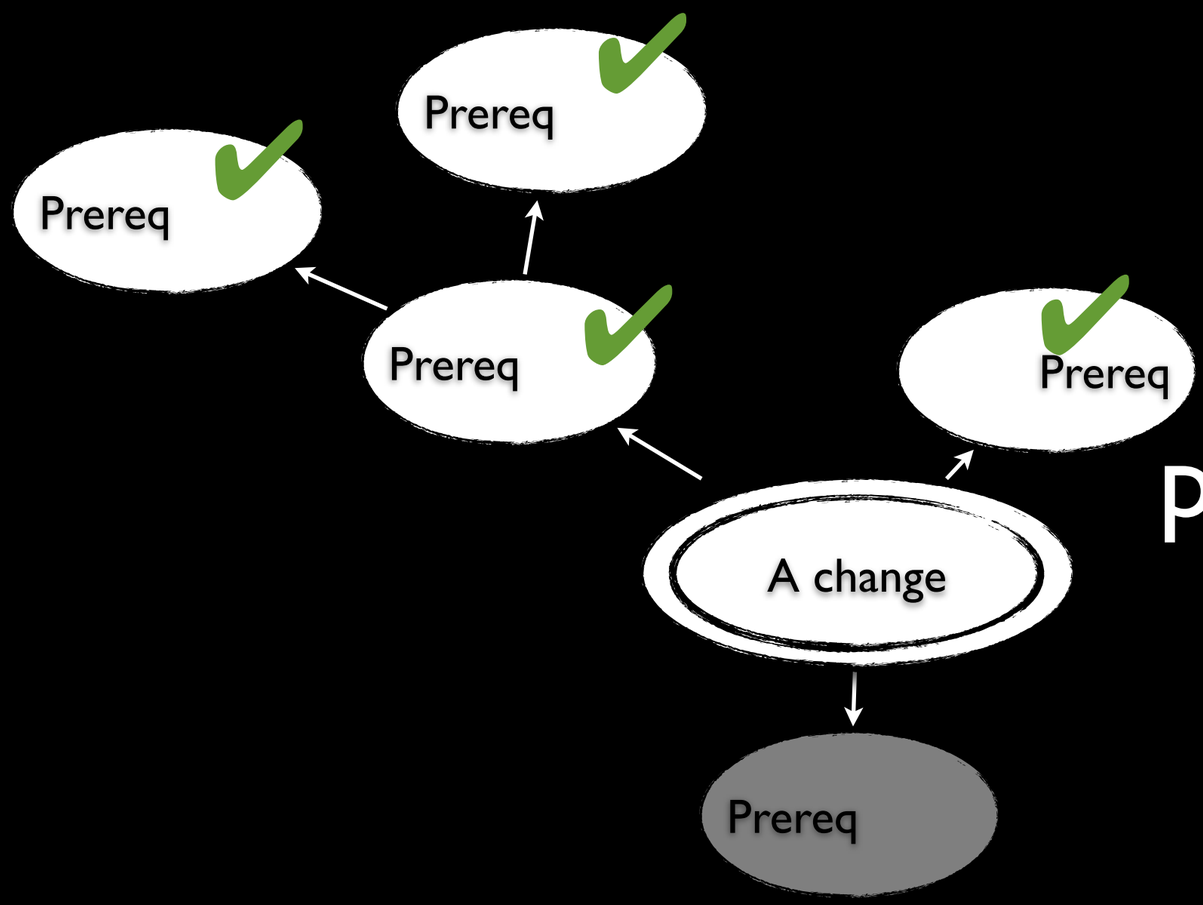


We continued with all leaves...

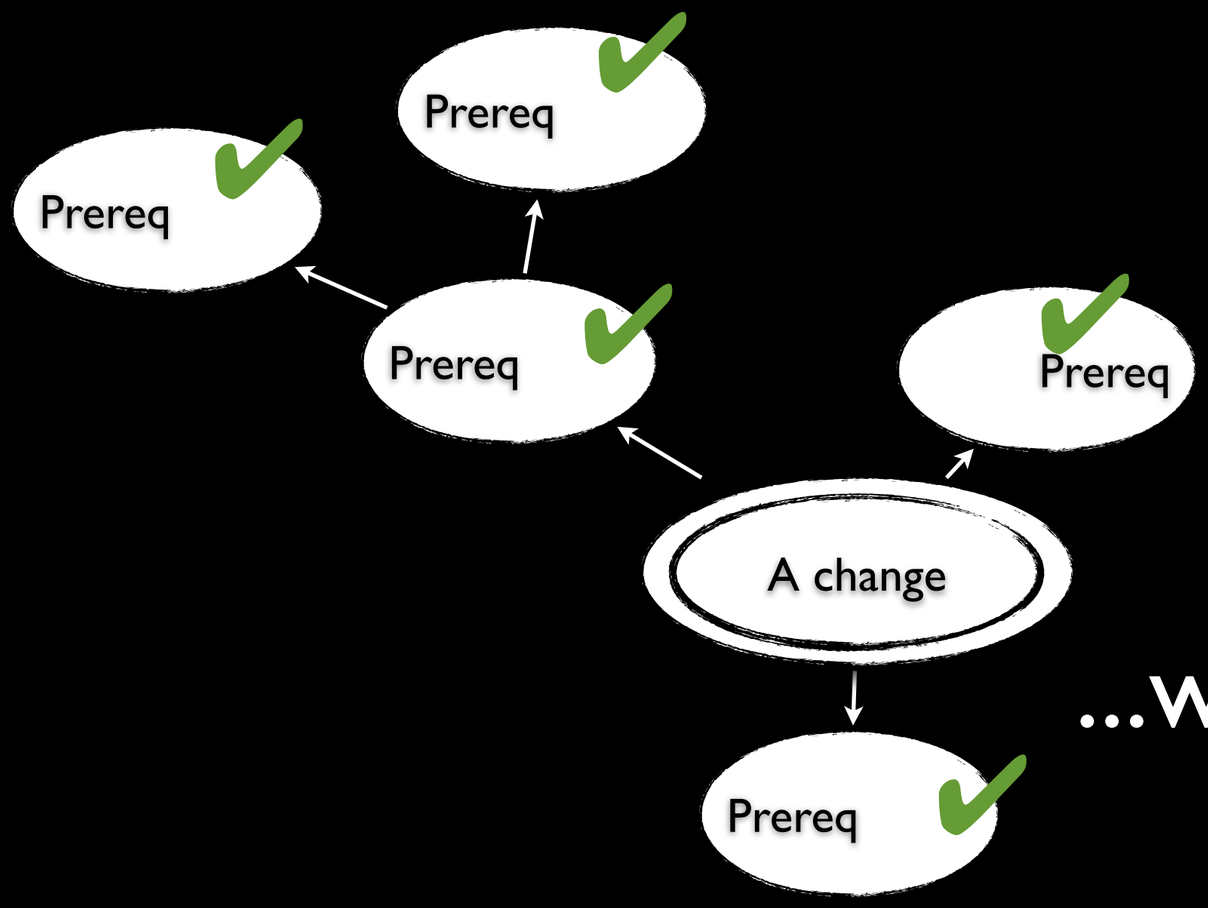


...getting transient prerequisites in place...

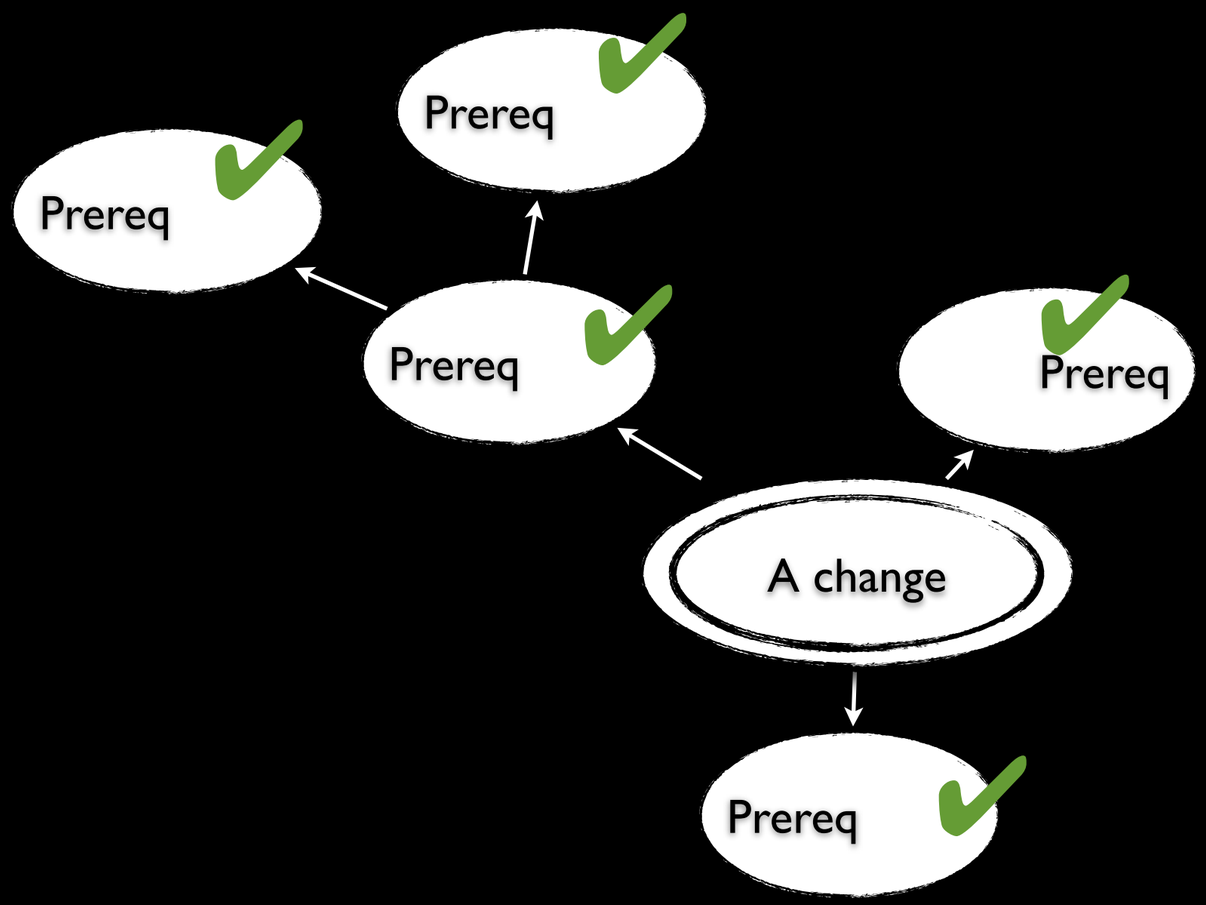




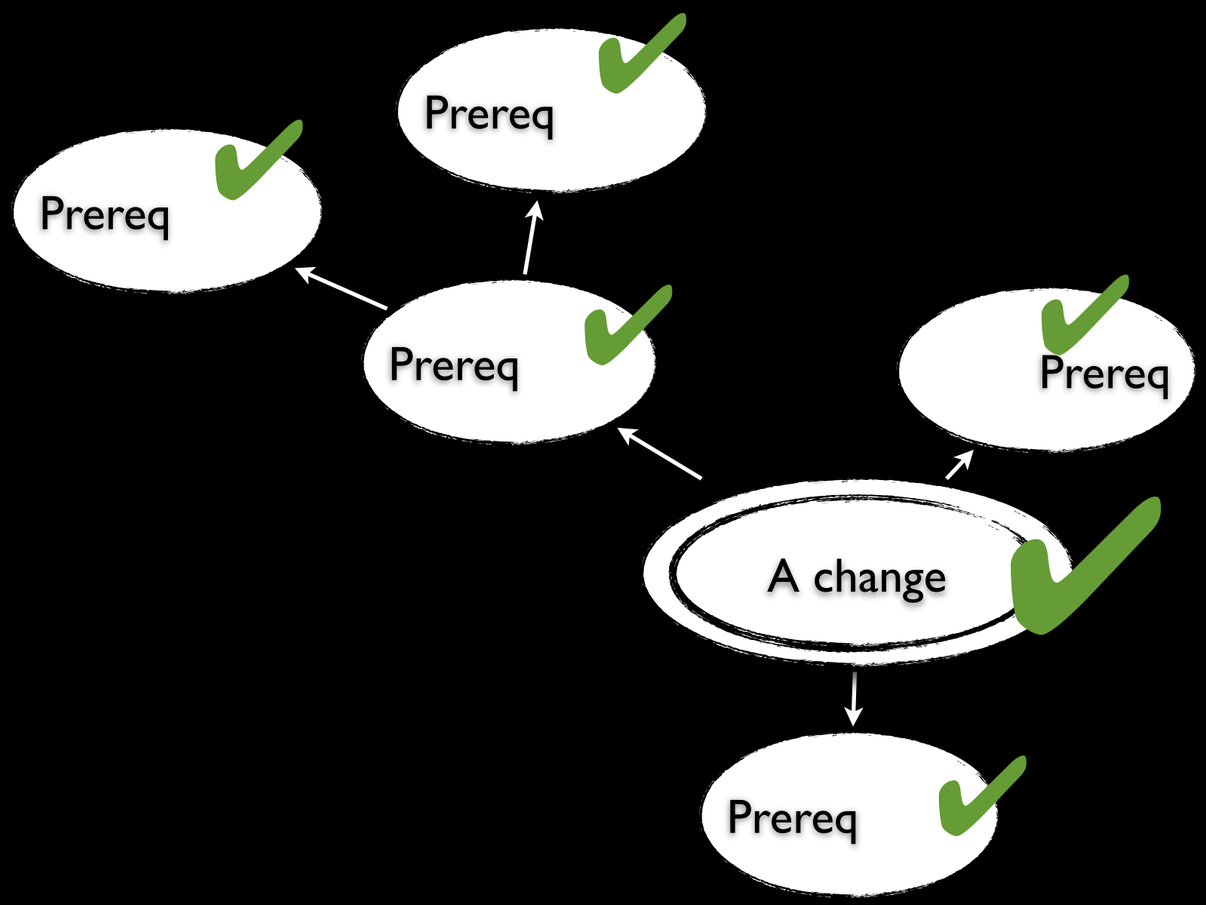
...fulfilling prerequisites...



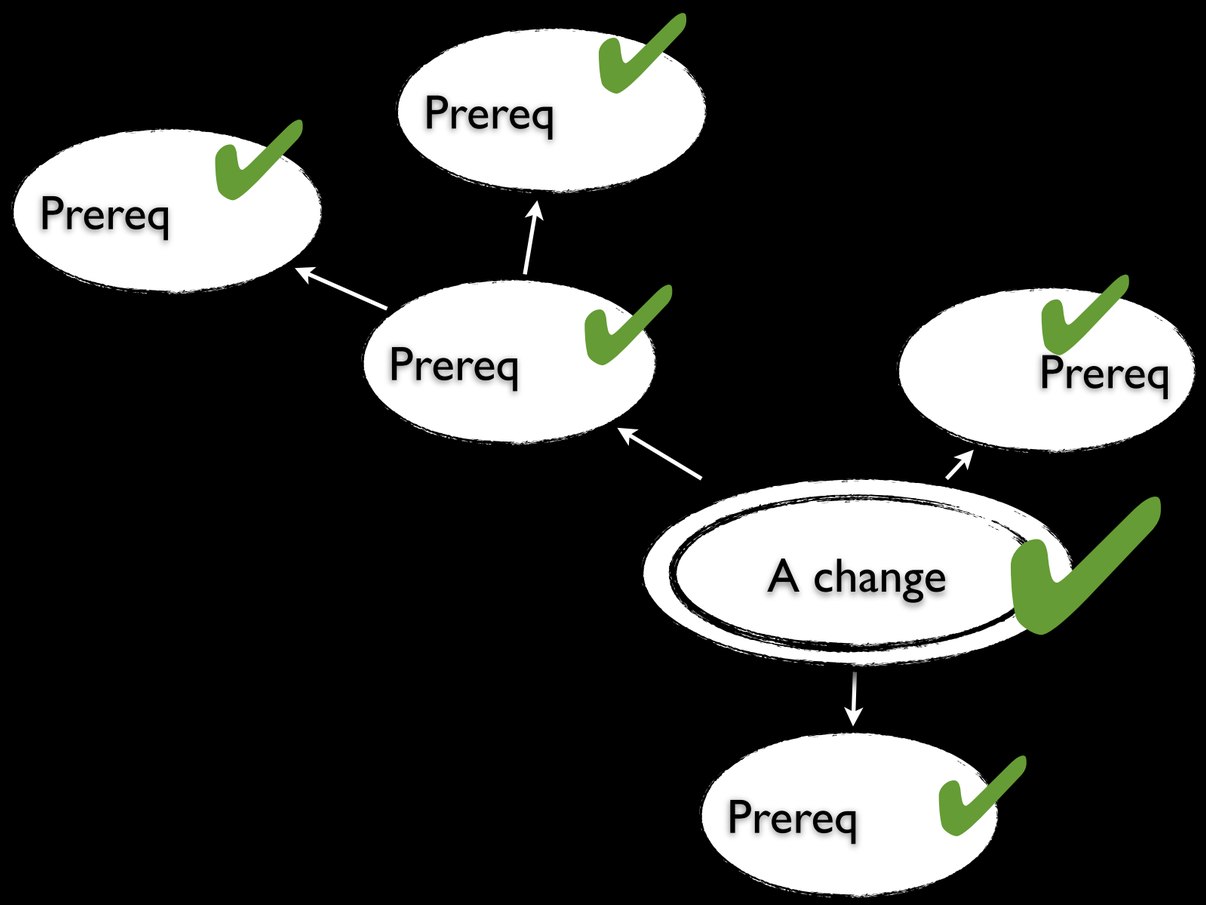
...working the way back to the original change



In time, all prerequisites
for the original change
were in place...



The change was now
easy to implement.



We're done!

Now, where's the Java code...?!?

Lets see an example!

https://github.com/brolund/mikadomethod/ (example1)

Welcome to Pasta Inc.

Recap

Benefits

Benefits

Always deliverable - from the main branch

Benefits

Always deliverable - from the main branch

Goal focus - do only the necessary

Benefits

Always deliverable - from the main branch

Goal focus - do only the necessary

Visualize - memo and cooperation

Questions?

Questions?

Is this instead of refactorings
or WELC?

What about dynamically typed languages?

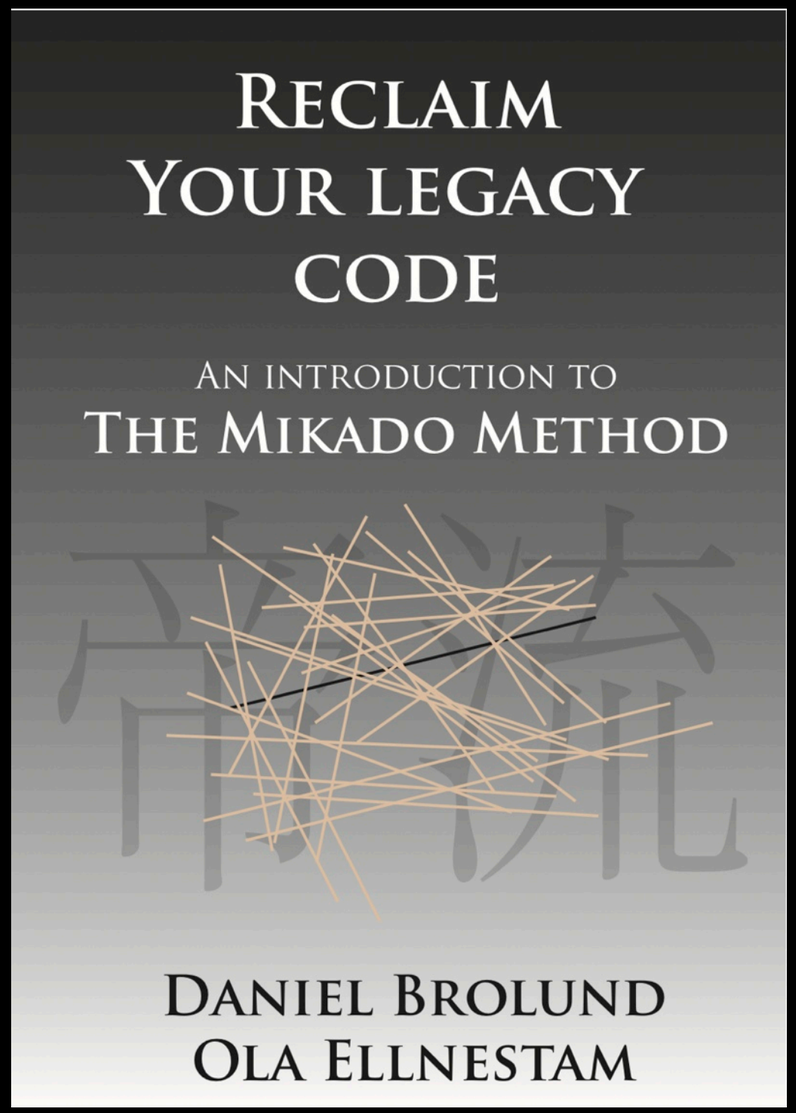
What about design principles?

How do I get started?

Thanks!!

Daniel Brolund @danielbrolund
Ola Ellnestam @ellnestam
Mikado Method @mikadomethod

http://mikadomethod.wordpress.com



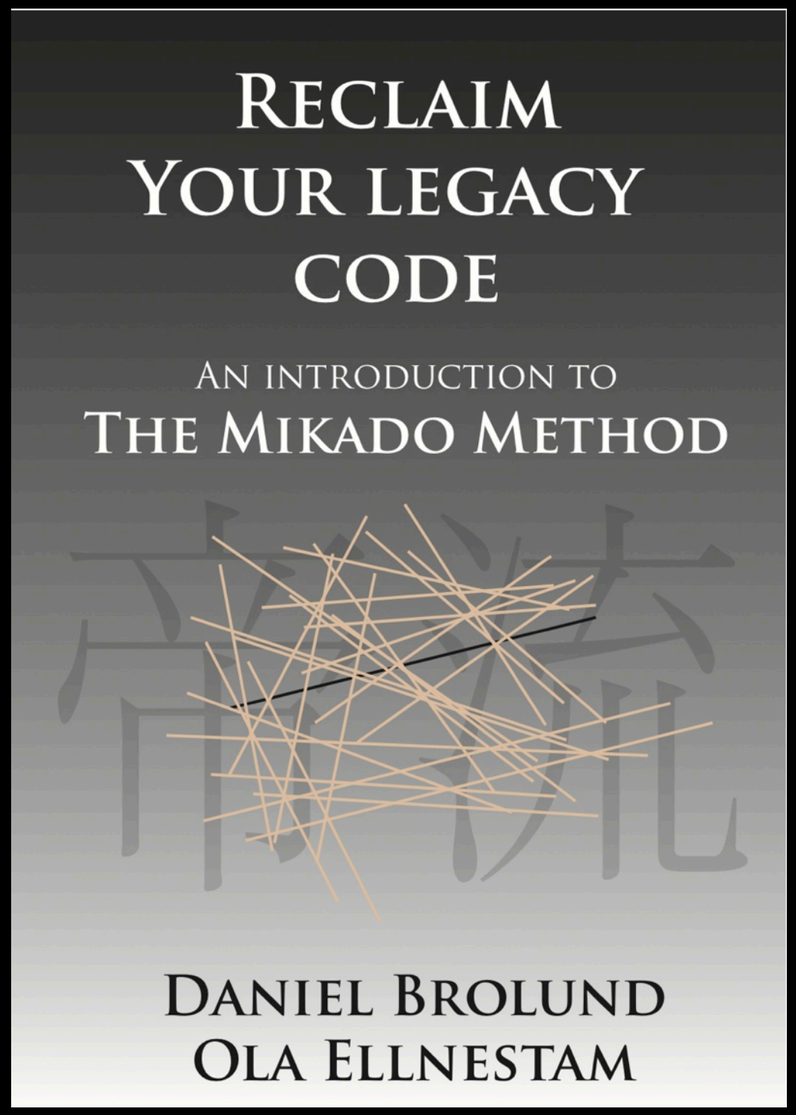
agical

ARE YOU MOVING AS FAST AS YOU CAN?

Thanks!!

Daniel Brolund @danielbrolund
Ola Ellnestam @ellnestam
Mikado Method @mikadomethod

http://mikadomethod.wordpress.com



agical

ARE YOU MOVING AS FAST AS YOU CAN?