SO MUCH MORE THAN…

A GRADLE BUILD MIGRATION BLUEPRINT

INTRODUCTIONS

- Chip Dickson
  - Chief Architect: SUM Global Technology
- http://sumglobal.com
- cdickson@sumglobal.com
- Twitter: @SUM_Global

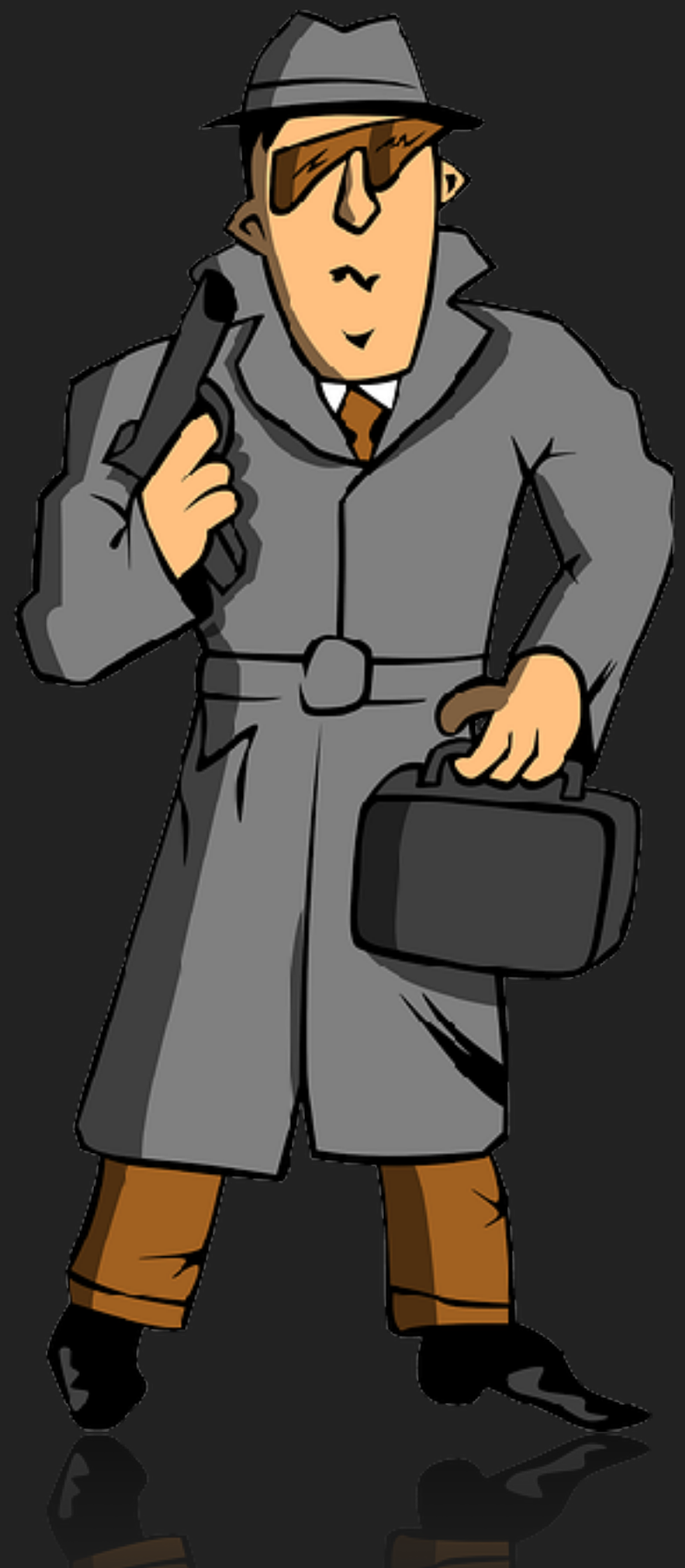
S·U·M
GLOBAL TECHNOLOGY

INTRODUCTIONS

- Charlie Walker
  - Sr. Architect SUM Global Technology
- http://sumglobal.com
- cwalker@sumglobal.com
- Twitter: @SUM_Global

THE ASSIGNMENT

- Fix our software
- Fix our build
- Fix our processes



S·U·M
GLOBAL TECHNOLOGY

FIX OUR SOFTWARE

- Why?
  - Production environment is unstable
  - Need to add new stuff but level of effort to do so is too large

FIX OUR BUILD

- Why?
  - Too complicated
  - Too slow
  - Non-repeatable
  - Others …

FIX OUR PROCESSES

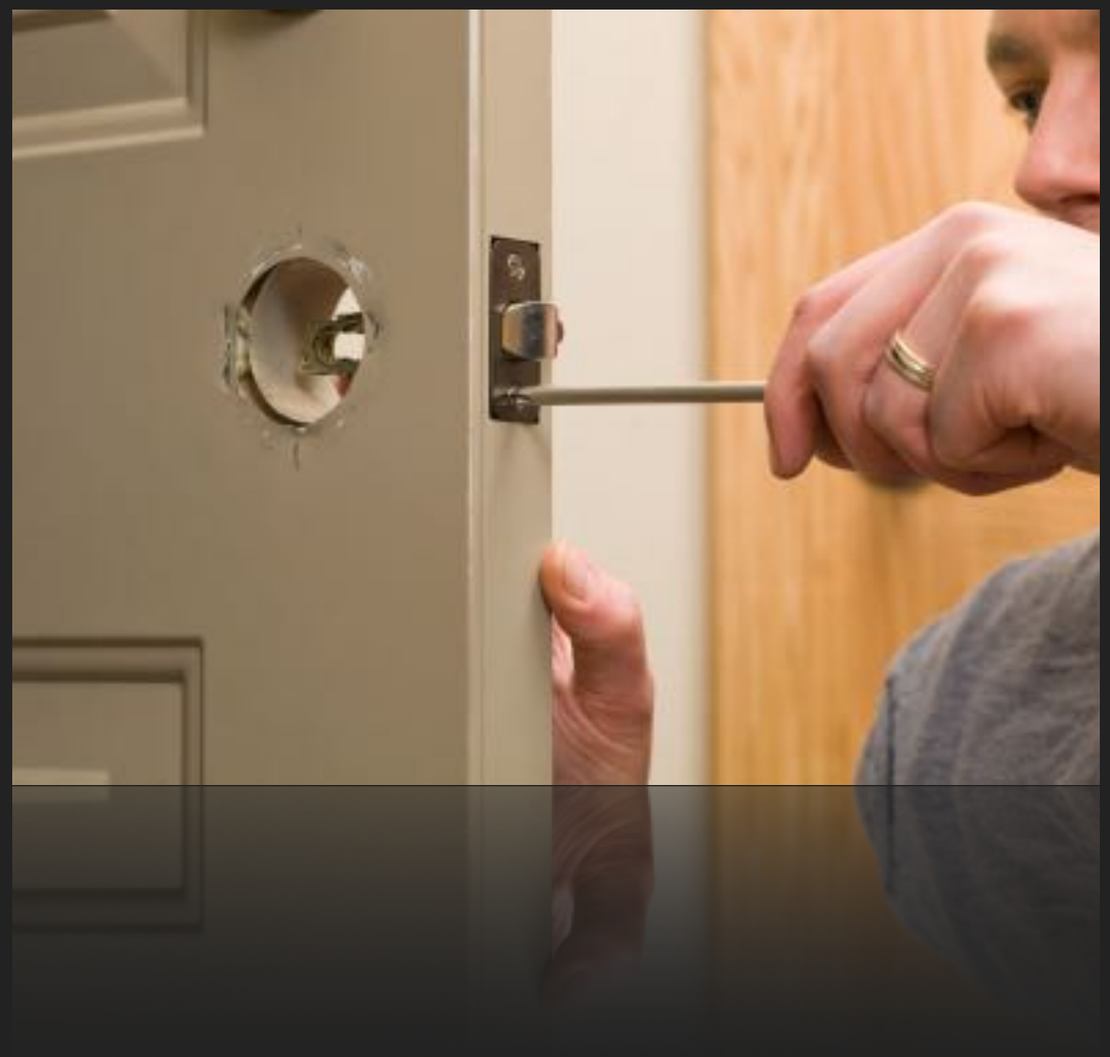
- Why?
  - Can’t properly audit build process
  - We don’t know what build is in production
    - Version
    - Source Files
  - We can’t test properly

TIME TO DIG A LITTLE DEEPER

S·U·M
GLOBAL TECHNOLOGY

WHAT THE CUSTOMER THINKS…

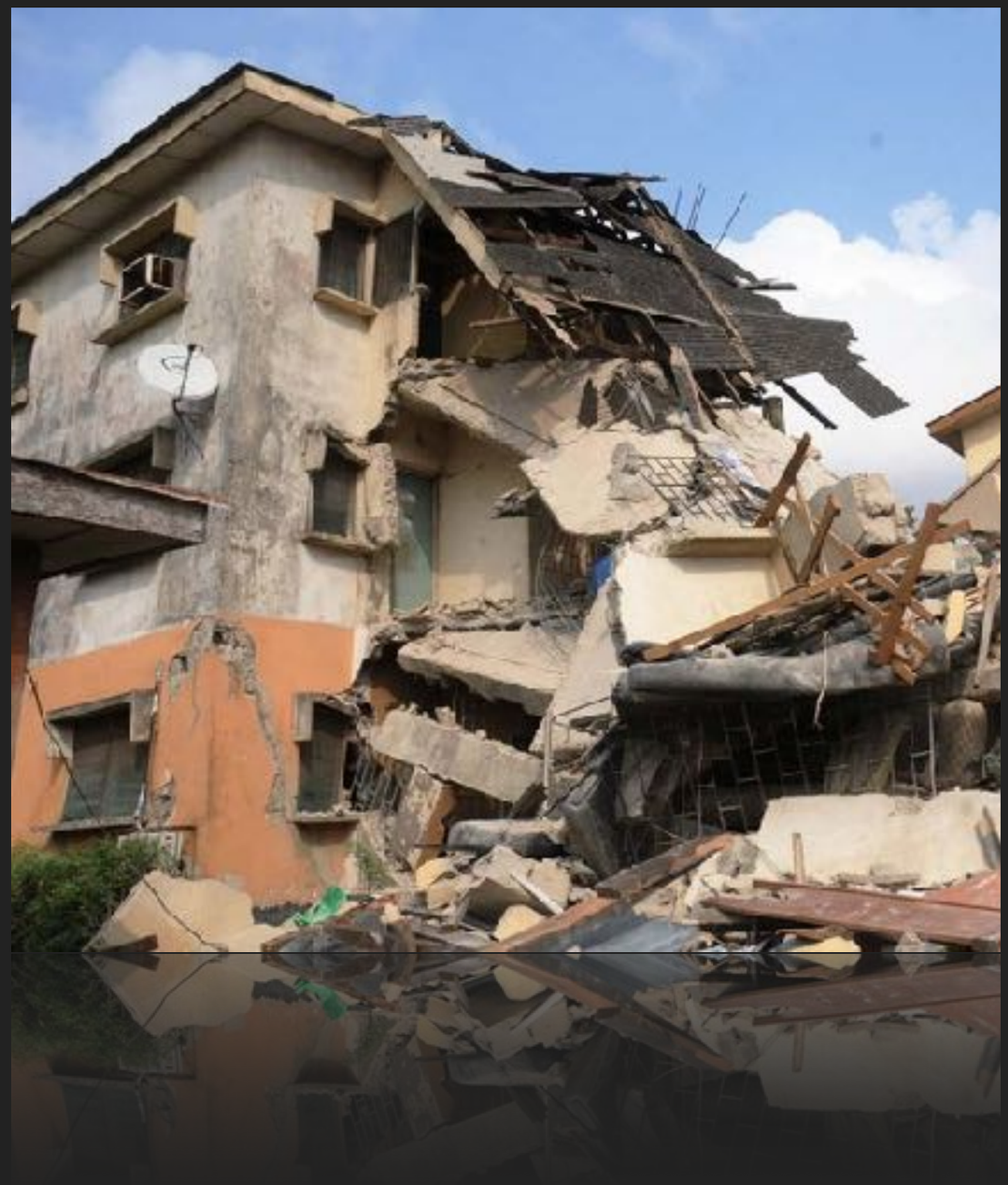
- Sr. Management believes they just need a little tweak here and there…



S·U·M
GLOBAL TECHNOLOGY

WHAT WE FIND . . .

- Things might be a bit worse than we were led to believe…



SOURCE CODE MANAGEMENT

- Goals
  - Traceability
  - Improved workflow
  - Improved organization

SOURCE CODE MANAGEMENT

- Do you have SCM?
  - Shared Folder
- Is the tool up to date?

SOURCE CODE MANAGEMENT

- How is the code organized?
  - Monolithic
  - Duplicate code
  - By deliverable

SOURCE CODE MANAGEMENT

- Is there proper versioning
  - Branching?
  - Tagging?

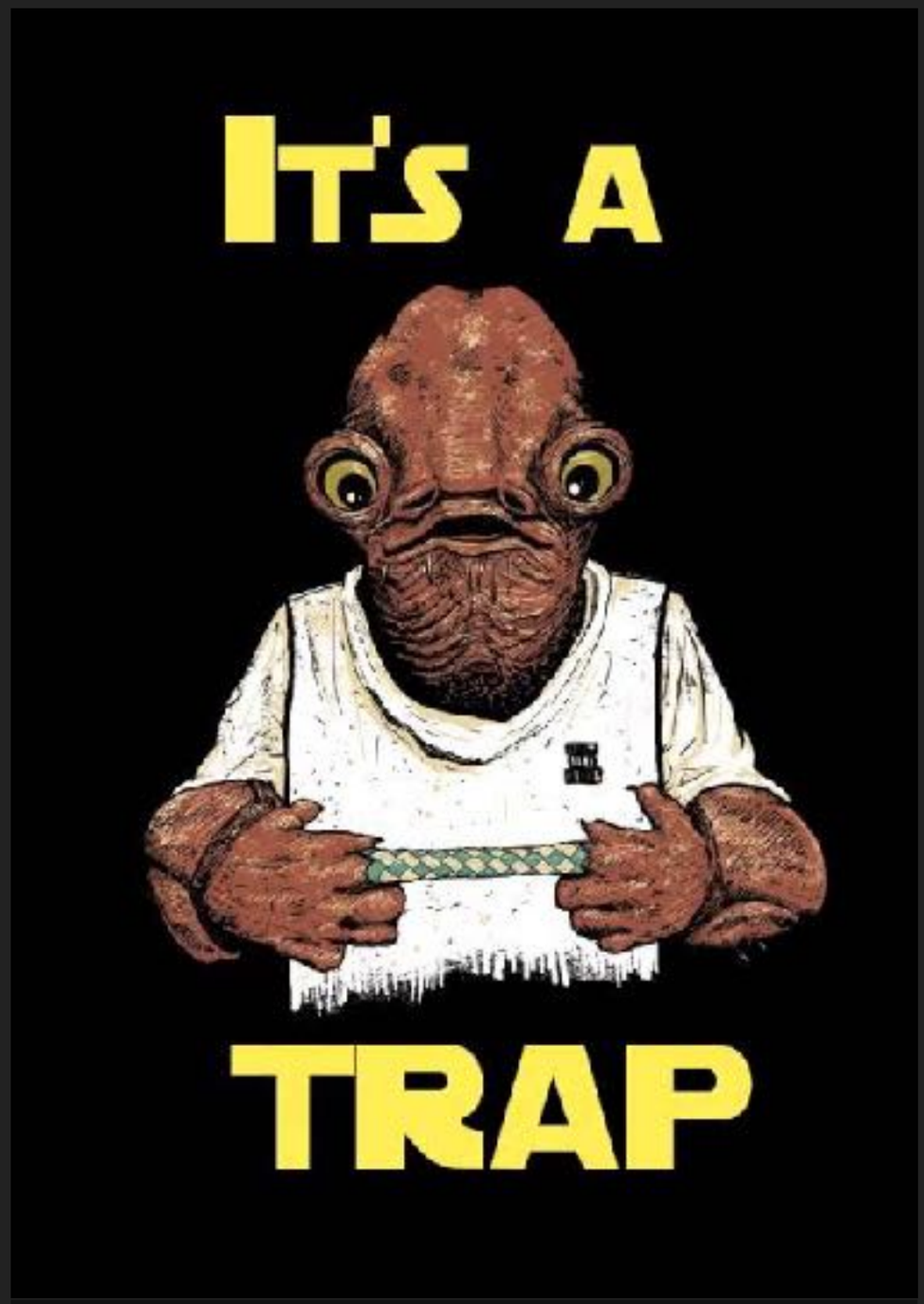
SOURCE CODE MANAGEMENT

- Check-In criteria
  - Proper comments?
  - Tied to Issue Tracking?

SOURCE CODE MANAGEMENT

- Introduce GIT
- Use a simple branching and tagging strategy
- All our problems are solved

SOURCE CODE MANAGEMENT



THE BUILD

- Goals
  - Easy to use
  - Performant
  - Organized
  - Dependency Management
  - Capable of supporting the process

THE BUILD

- Most common build tool we find?
  - IDE
- Most common dependency management?
  - “lib” folder per project
- Most common artifact repository?
  - Shared folder

THE BUILD

- Most common troubleshooting comment heard with this tool stack?

S·U·M
GLOBAL TECHNOLOGY

THE BUILD

THE BUILD

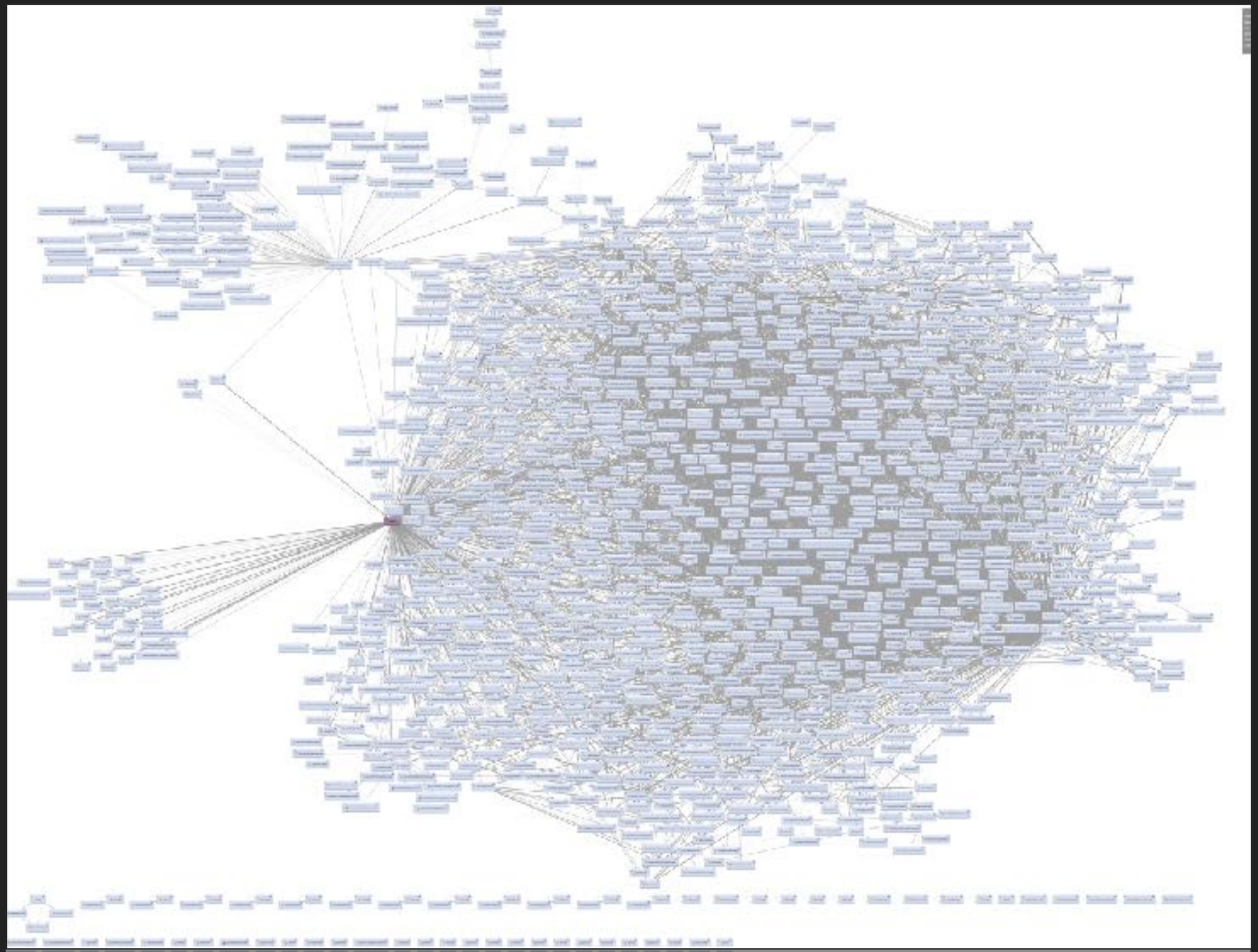
- Other build tools we find
  - Ant
    - Your grandfather's build system
  - XML Configuration
    - NOT!
- Still a first class citizen in Gradle

THE BUILD

- Seriously??
  - More XML??
  - Oh, but at least its less flexible!
- Just say no to Maven…

THE BUILD

- Dependency management



THE BUILD

- Manual dependency (/lib)
  - Determining library versions
  - Duplicates across projects and conflicts
  - Determine runtime vs compile time vs testing dependencies

THE BUILD

- war, jar or what?
  - What is the deployable artifact?
    - WAR...
      - What is it good for?
    - rpm/deb
    - Docker

THE BUILD

- Artifact Storage
  - Its built… now what?
  - Put it back in the shared folder?
    - uhhh no…
  - Artifact storage repository
    - No, not SVN or GIT
    - Nexus, Artifactory are good options
- Store intermediary, final source code artifact and the deployment artifact

THE BUILD BIG PICTURE

- What we solved so far...
  - SCM - GIT
    - Project layout
    - Branching and Tagging
  - Build Script
    - Gradle
  - Artifact Repository

THE BUILD BIG PICTURE

- What we have left to discuss…
  - Build is still running on a laptop, just not in an IDE
  - Projects are not organized for efficiency
    - Still possible duplicate code
  - Issue Tracking and traceability

THE BUILD

- Continous Integration
  - What is the production environment?
  - Artifacts move through the environments
  - Vagrant, Docker

THE BUILD

- Continous Integration
  - Build Artifact Repository
    - Email
    - Shared drive/mount
    - Artifactory, Bintray, Nexus
    - YUM, APT
    - Docker Hub / Registry

THE BUILD

- Build Server
  - Jenkins
    - There are others
  - Build on commit
    - Incremental (snapshot)
- Build Nightly
  - Clean build (snapshot)
- Release Builds

THE BUILD

- Pitfalls
  - Resistance to change
    - “We have always done it this other way…. why do we need to change?
  - Your team recognized something wasn’t working
    - Called in for a reason

S·U·M GLOBAL TECHNOLOGY

THE BUILD

- Pitfalls
  - This much change is too RISKY!
    - Can’t execute all this and stay productive
    - Point out the risk of not changing
  - Usually points back to why we were asked to help in the first place

THE BUILD

- Pitfalls
  - Not built here!!!
    - “We didn’t think of it, so it can’t be the correct solution”
  - Two approaches we have taken
    - Guided Discovery
    - Fresh Eyes on the problem

THE BUILD

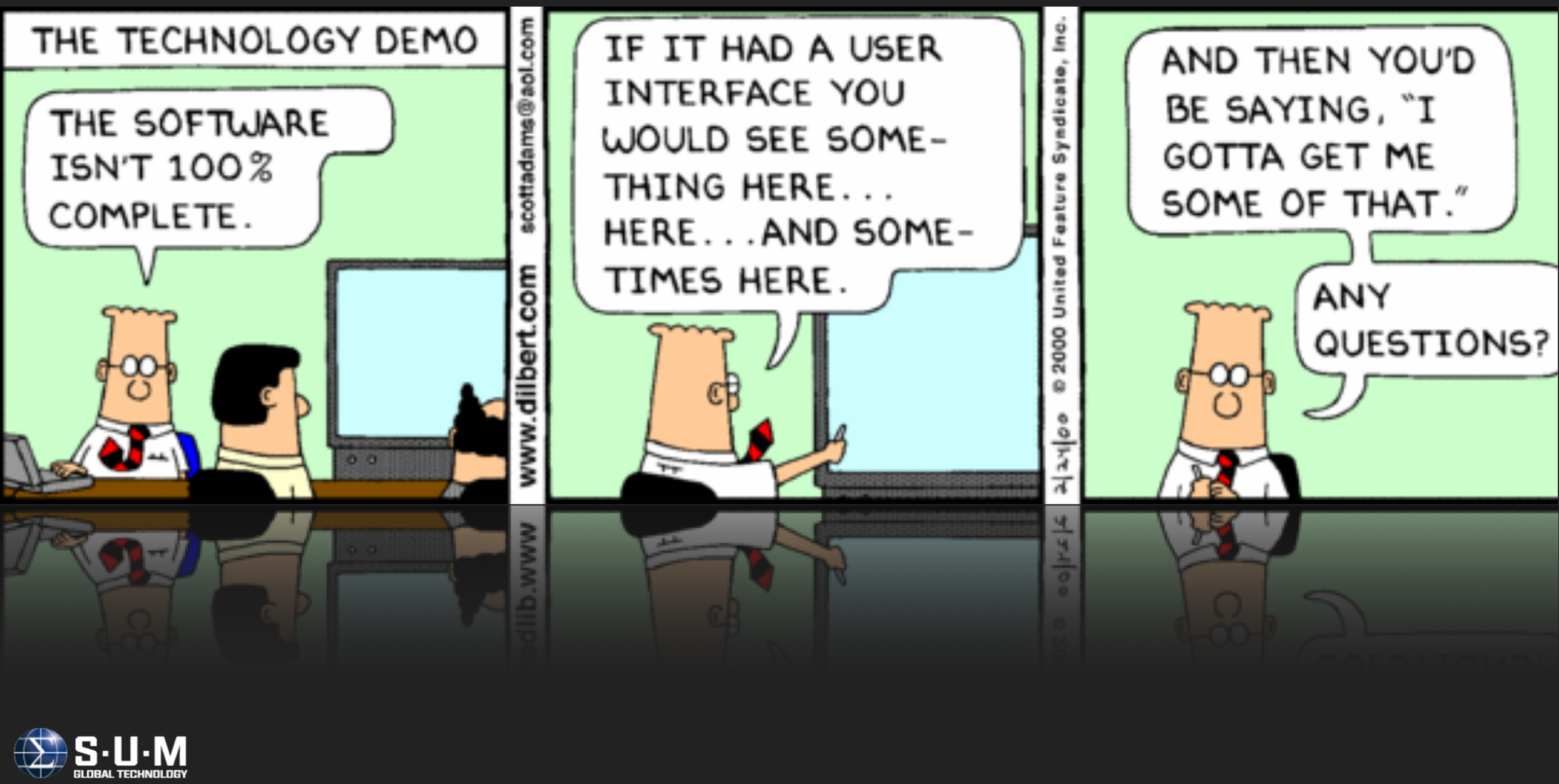
- Pitfalls
  - Too Expensive
    - “We can’t afford to make all these changes”
  - Opportunity cost of not fixing
- Point out actual costs associated with the current solution
- Point out savings from the upgraded solutions

S·U·M GLOBAL TECHNOLOGY

THE BUILD

- Summary
  - GIT - SCM
  - Gradle - Build technology
  - Nexus - Artifact repo
  - Jenkins - Build server
  - Docker - deployment
    - Vagrant if Docker is too much

DEMO


THE TECHNOLOGY DEMO
THE SOFTWARE ISN'T 100% COMPLETE.
IF IT HAD A USER INTERFACE YOU WOULD SEE SOMETHING HERE... HERE...AND SOMETIMES HERE.
AND THEN YOU'D BE SAYING, "I GOTTA GET ME SOME OF THAT."
ANY QUESTIONS?
scottadams@aol.com
www.dilbert.com
© 2000 United Feature Syndicate, Inc.