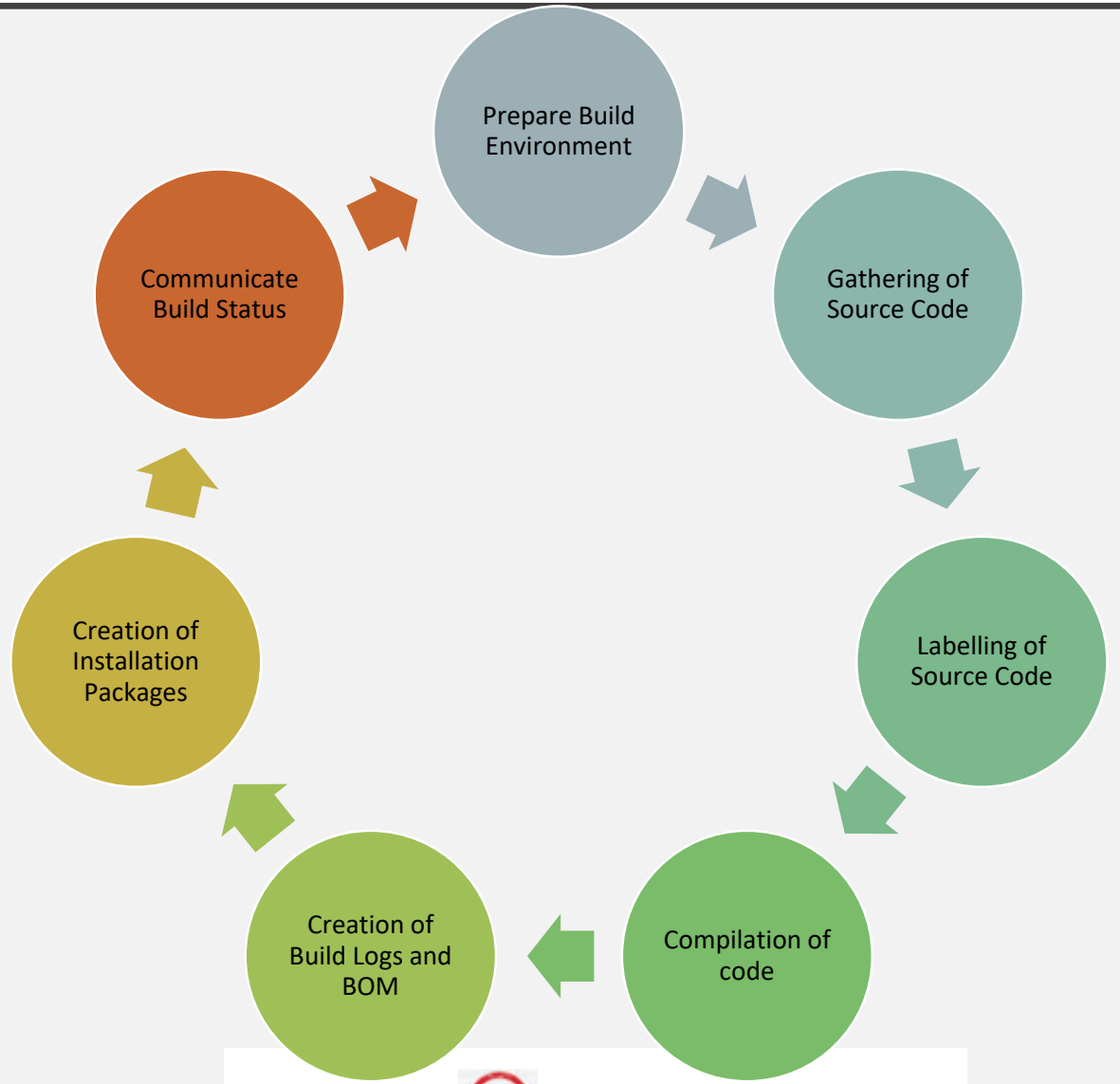
INTRODUCTION TO CORE BUILD MANAGEMENT

WHAT IS BUILD MANAGMENT

BUILD MANAGEMENT DEFINED

- Build Management is the process of assembling **all** the components of a software application into an **installable software product**.

A TYPICAL CYCLE



SYSTEM INTEGRATION AND BUILD MANAGEMENT

OUTLINE

- Continuous Integration
- Continuous Builds
- Continuous Tests
- Tools

CONTINUOUS INTEGRATION

CONTINUOUS INTEGRATION

- “Traditional” Integration methods
  - Integration before the end of iteration
  - Weekly integration

- “Big Bang” Integration
  - Too many bugs and too little time to fix them

CONTINUOUS INTEGRATION

- CI principles
  - Integrate more often: Daily, Hourly
  - Integrate continuously:
    After every completion of a task

CONTINUOUS INTEGRATION

- Advantages
  - Integration becomes trivial if you integrate small and simple tasks
  - Developers get a better overview of the system and its architecture
  - In turn, integration becomes easier

CONTINUOUS BUILDS

CONTINUOUS BUILDS

- CB principles
  - Build your changes locally before checking in
  - If it doesn't compile, don't check it in
  - Build the entire system "from scratch" on a dedicated build server

CONTINUOUS BUILDS

- Advantages
  - Reduces the problems with files that aren’t checked in
  - Reduces problems with debug and release versions
  - In a integrated and stable stage builds can serve as demo-versions

CONTINUOUS TESTS

CONTINUOUS TESTS

- CT principles
  - Test your changes locally before checking in
  - If it doesn't pass the tests at 100%, don't check it in
  - Start the system and run the full test-suite

CONTINUOUS TESTS

- Advantages
  - Reduces bugs
  - Increases the chance to have a running system at every time

CONTINUOUS TOOLS

BUILD TOOLS

- Cruise Control
- Ant
  - Pure build tool
- Maven
  - A lot functionality “out of the box”
- MSBuild Engine

TEST TOOLS

- Maven
  - Runs xUnit test suites
- xUnit
  - Suite of Test Frameworks for several languages (Java, C, C++, Perl, ….)

INTEGRATION TOOLS

- Subversion
- CVS

BEST PRACTICES

RULE #1: DEDICATED BUILD SERVERS

- Eliminates unauthorized access to make changes in the server configuration
- Reduces the risk the creation of unknown build dependencies
- Reduces build time because the build is not fighting other processes and application for server resources

RULE #2: FULLY AUTOMATE THE PROCESS

- Ensures repeatability and reproducibility of the build process
- Eliminates risk of human error in the build process

RULE #3: DAILY BUILDS

- Ensures that the full system is always in a build-able state
- Allows for daily test runs on the daily build

RULE #4: CI BUILDS

- Protects that the Daily full system build will complete successfully
- Immediate success or failure feedback of developer check-ins

RULE #5: RUN UNIT/SMOKE TESTS

- In addition to ensuring the system is in a build-able state, this ensures the system's runtime works

RULE #6: BUILD IN CLEAN WORKSPACE

- Ensures no unexpected build or runtime dependencies are created

RULE #7: CHECK IN ALL FILES

- Ensures that the required source and build files are properly under source code control
- Helps with build reproducibility

RULE #8: EACH BUILD HAS UNIQUE IDENTIFIER

- Ensures every build is repeatable and reproducible

RULE #9: CREATE A BOM

- BOM, bill of materials, of the source code versions for each build
  - Visual confirmation of the source code versions in the build
  - Aids in developer debugging of builds

RULE #10: CREATE BUILD MANIFEST

- Visual confirmation of build artifacts

RULE #11: MAINTAIN BUILD LOGS

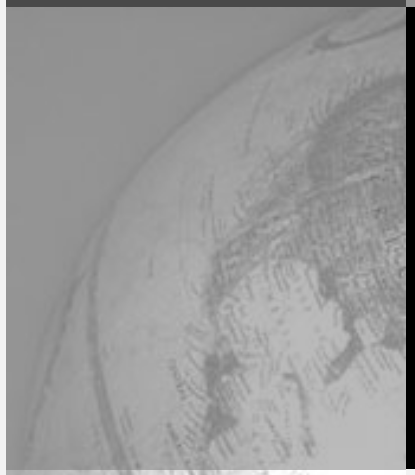
- Builds an audit trail
- Debugging tool for builders and developers

RULE #12: COMMUNICATE

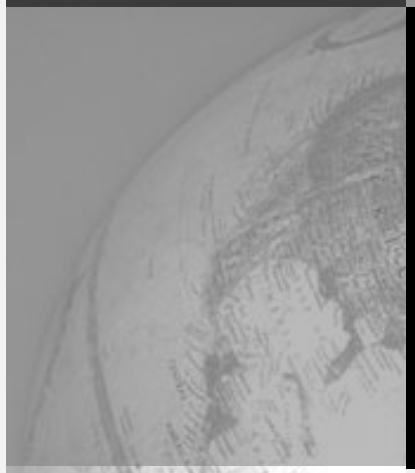
- Proactive build notification of success or failure

RULE #13: TRACK RELEASE BASELINES

- Ensures reproducibility of software release



Questions and Answers.



Thanks for listening!!!

Seshagiri Sriram
Email: SeshagiriSriram@gmail.com