UECS2363 SOFTWARE CONSTRUCTION AND CONFIGURATION
CHAPTER 7 : CONTINUOUS INTEGRATION (CI)

DR FARIZUWANA AKMA
farizuwana@utar.edu.my

What is Continuous Integration?

"… a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily --- leading to multiple integrations per day. Each integration by an automated build (including test) to detect integration errors as quickly as possible. …"

-- Martin Fowler

Implementing CI

- What you need before you start:
  - Version control
  - An automated build
  - Agreement of the team

Version Control

- Everything in the project must be checked in to a single version control repository:
  - Code
  - Tests
  - Database scripts
  - Build and Deployment scripts
  - Anything else to create, install, run and test your application

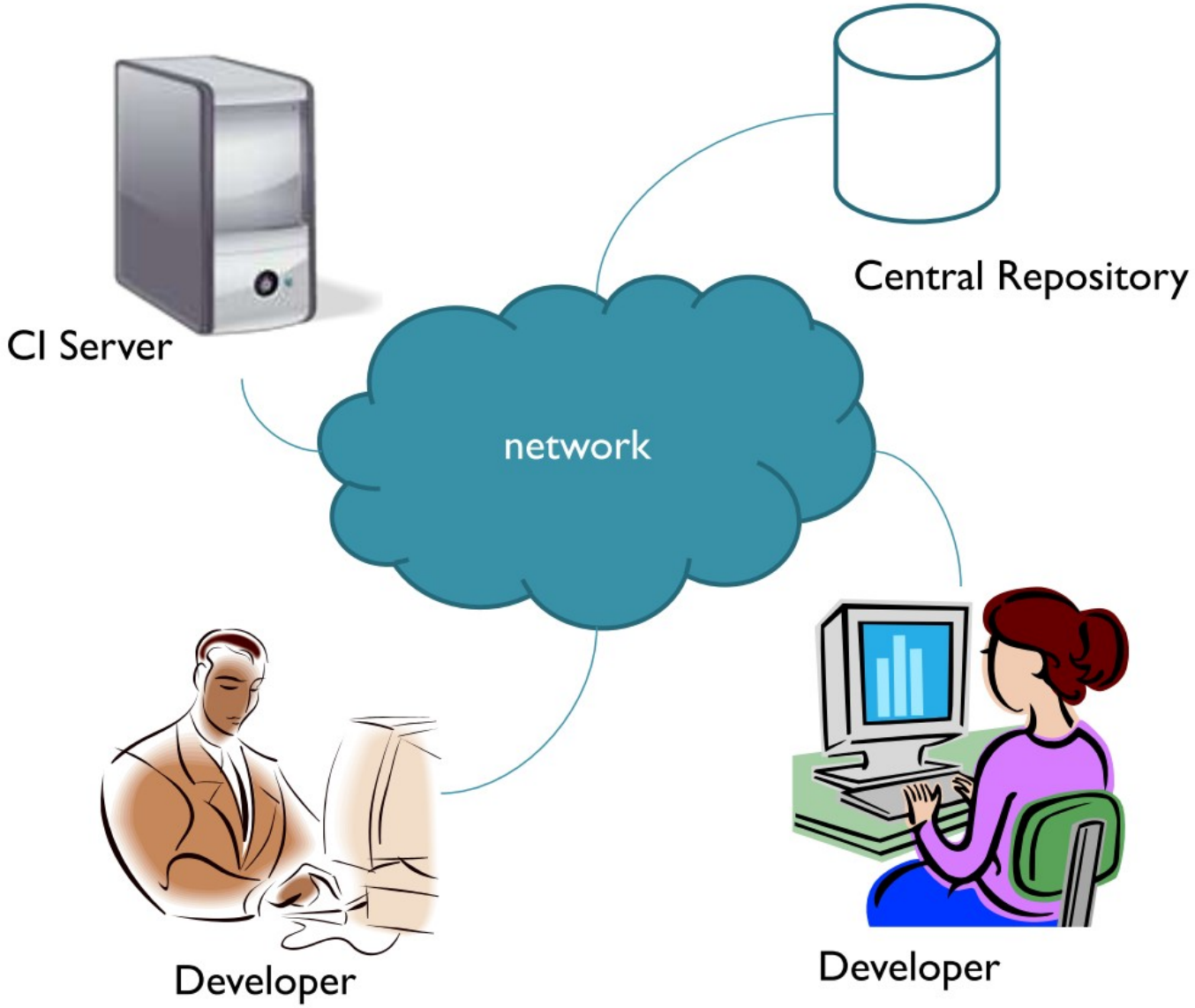
An Automated Build

- Must be able to start your build from the command line
  - build process must be automated from your CI environment so that it can be audited when things go wrong
  - Build scripts should be treated like codebase
  - It makes understanding, maintaining and debugging the build easier, and allows for better collaboration with operations people

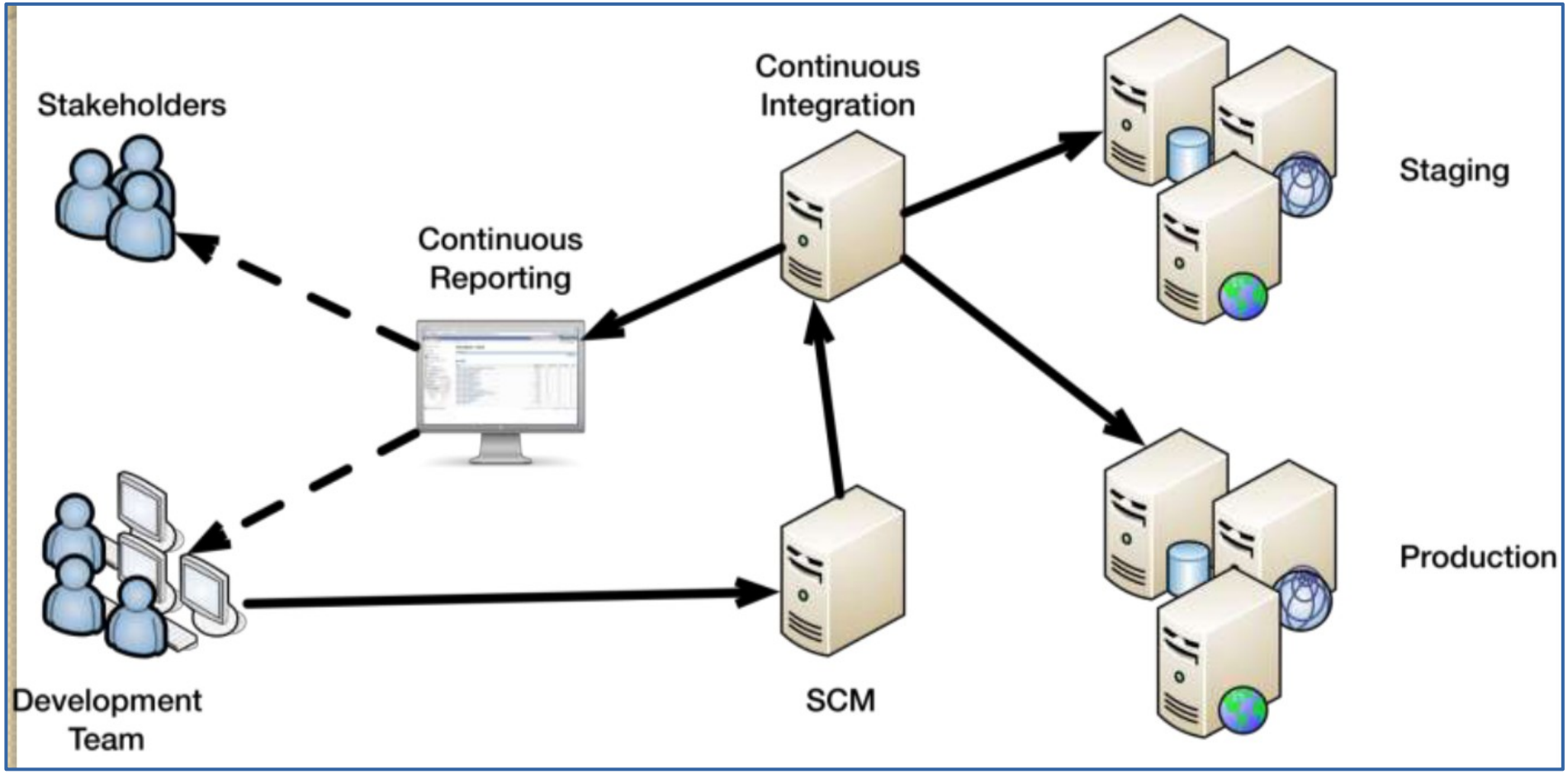
Agreement of the Team

- CI is a practice, not a tool
- Requires a degree of commitment and discipline from the development team
- Everyone check in small incremental changes frequently to mainline
- Agree that the highest priority task on the project is to fix any change that breaks the application

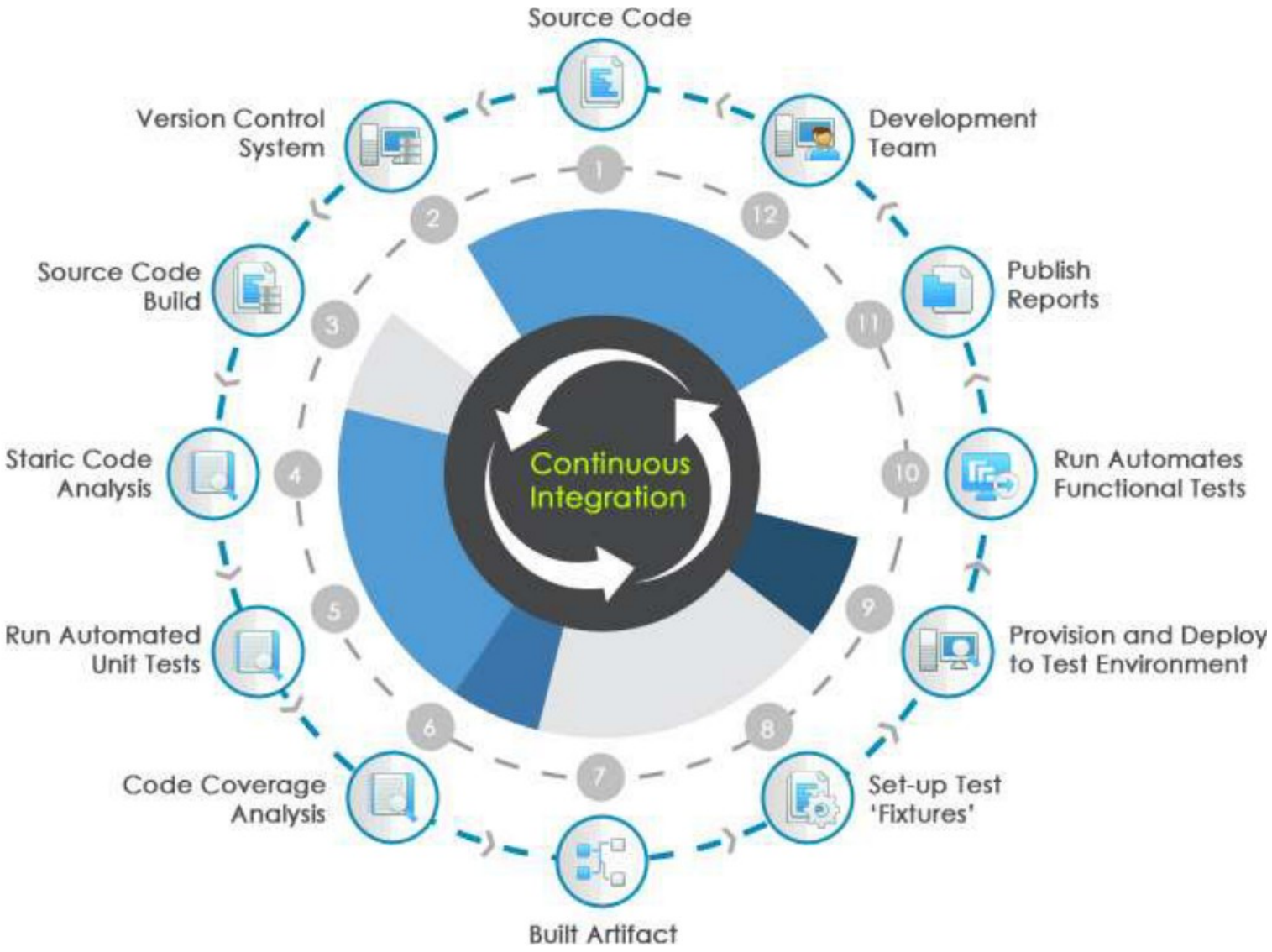
Basic CI Setup



Continuous Integration Workflow



CI Process Cycle



CI Tools

- CI tools are extremely simple to install and get running
- Open source options:
  - Jenkins / Hudson
  - CruiseControl
  - Gitlab CI
- Commercial
  - Go from ThoughtWorks Studios
  - TeamCity from JetBrains
  - Travis CI

Pre-requisites for CI

- Check In Regularly
- Create a comprehensive automated test suite
- Keep the build and test process short
- Managing your development workspace

Check In Regularly

- **Check in regularly to trunk or mainline of the version control system, i.e. at least a couple of times a day**

Create a comprehensive automated test suite

- **Unit tests**
  - **test the behavior of small pieces of you application in isolation.**
  - **Don’t require your application to be running in a productionlike environment**
  - **Do not hit the database, file systems or other systems.**
  - **Should run very fast, i.e. less than 10 minutes even for large system**

Create a comprehensive automated test suite

- **Component tests**
  - test the behavior of several components of your applications.
  - Don’t always require starting the whole applications
  - May hit database, file systems or other systems
  - Typically take longer to run than unit tests
- **Acceptance tests**
  - Test that the application meets the acceptance criteria decided by the business

Keep the build and test process short

- If the build and test process is long, then the team will face the following problems:
  - People will stop doing a full build and running the tests before they check in
  - The CI process will take so long that multiple commits will have taken place by the time you can run the build again, so you won't know which check-in broke the build
  - People will check in less often

Keep the build and test process short

- Ideally the compile and test should be < 10 minutes
- Contradicting to the previous one – having a comprehensive set of automated tests
- To reduce build times
  - Make your test run faster, analyze and optimize tests
  - Split your test process into multiple stages

Essential Practices for CI

- Don’t check in on a broken build
- Always run all commit tests locally before committing, or get your CI server to do it for you
- Wait for Commit tests to pass before moving on
- Never go home on a broken build

Essential Practices for CI

- Always be Prepared to Revert to the Previous Revision
- Time-box Fixing before reverting
- Don’t comment out failing tests
- Take responsibility for all breakages that result from your changes
- Test-driven Development

Managing your development workspace

- Developers should carefully manage their development environment for productivity
- Should be able to perform these tasks under their control on their own local machine:
  - To run the build
  - Execute the automated test
  - Deploy the application

Managing your development workspace

1) Careful configuration management
   - Source code, test data, database scripts, build scripts, deployment scripts, etc.
2) Configuration of third-party dependencies, libraries, and components.
3) Make sure the automated test can be run on developer machines.

SUMMARY

Continuous Integration

- **Continuous integration (CI) is a practice in software engineering of merging all developer working copies with a shared mainline frequently. Its main aim is to prevent integration problems.**

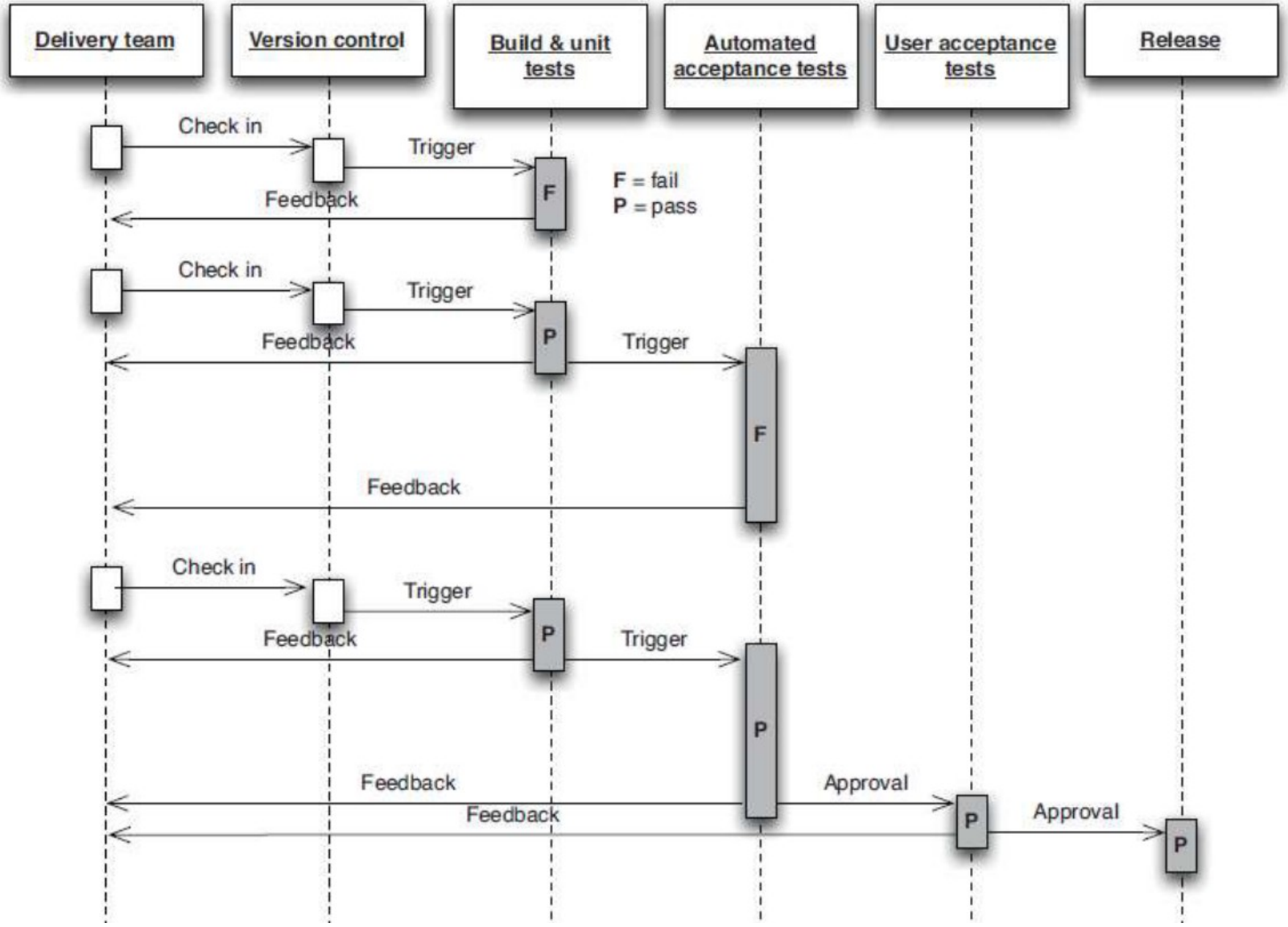
Benefits of CI

- CI allows early detection of errors in the codes
- CI removes integration sessions
- CI minimizes the integration bugs
  - If you build and test your software once an hour, no problem is more than an hour old.
- CI improve team works
- CI delivers latest best build products
- CI reduces overall development cost
  - making it easier to find and fix problems
  - provides valuable and timely information, letting the development be managed more tightly.

CI Success Factors

1) Single source code repository
2) Automated build scripts
3) Automated tests
4) Developers' disciplines
   - Synchronize often
   - Don't break the build
   - When you break the build, fix it

Deployment Pipeline



END OF LECTURE 10