Continuous Integration

Agenda

- Definition
- CI Considerations
- About tools
- CI as a scheduling in the development process

Sources

- Duvall, Matyas and Glover. Continuous integration improving software quality and reducing risk. Addison Wesley, 2007.
- Martin Fowler. Continuous Integration. Available in http://www.martinfowler.com/articles/continuousIntegration.html
- Schach, Stephen R. Object-Oriented and Classical Software Engineering Eighth Edition. Mc Graw Hill, 2010
- And so *others*…

Critical Issue

Supporting the idea of creating systems to continuously build and test systems in response to changes in source control.

Continuous Integration

(From Wikipedia)

- Continuous Integration emerged in the ***Extreme Programming*** (XP) community
- XP advocates Martin Fowler and Kent Beck first wrote about continuous integration (1999).
- Beck's book ***Extreme Programming Explained***, the original reference for Extreme Programming, also describes CI.
- Fowler's paper is a popular source of information on the subject.

Martin Fowler and CI

In his very popular "Continuous Integration" article, Martin Fowler describes CI as:

- *". . . 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 is verified by an **automated build (including test)** to detect integration errors as quickly as possible.*
- *Many teams find that this approach leads to significantly reduced integration problems and **allows a team to develop cohesive software more rapidly**"*

Wikipedia say us!!!

Continuous integration (CI) implements continuous processes of applying ***quality control*** — small pieces of effort, applied frequently.
Continuous integration aims to ***improve the quality of software***, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control after completing all development.

From: http://en.wikipedia.org/wiki/Continuous_integration

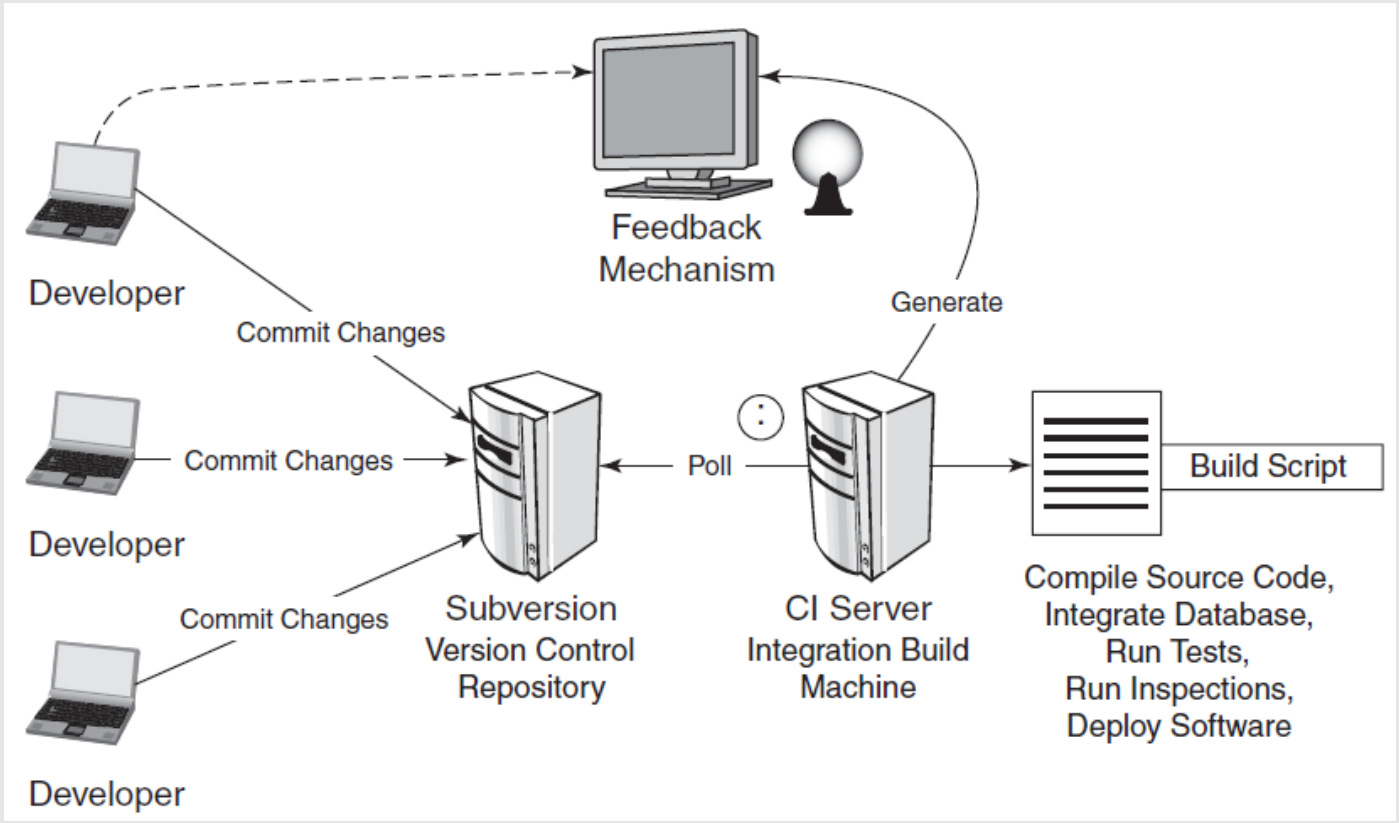
This means

- Continuous integration improving software quality and reducing risk
- As the complexity of a project increases (even just adding one more person), there is a greater need to integrate and ensure that software components work together—early and often.
- Waiting until the end of a project to integrate leads to all sorts of software quality problems, which are costly and often lead to project delays. CI addresses these risks faster and in smaller increments

The value of CI is to:

- Reduce risks
- Reduce repetitive manual processes
- Generate deployable software at any time and at any place
- Enable better project visibility
- Establish greater confidence in the software product from the development team

Typical CI context



From: Continuous integration improving software quality and reducing risk

Beyond to the CI Tools…

Practicing CI is more than installing and configuring some tools.

- *How many of the following items are you consistently performing on your project?*
- *How many of the other CI practices can improve your development capabilities?*

Beyond to the CI Tools…

- On average, is everyone on your team committing code at least once a day? Are you employing techniques to make it easier to commit code often?
- What percentage of each day's integration builds is successful (that is, the most recent build run has passed)?
- Is everyone on your team running a private build before committing to the repository so that integration errors are reduced?
- Have you scripted your builds to fail if any of your tests or inspections fail?
- Is a broken integration build a priority to fix on your projects?
- Do you avoid getting the latest code from the version control system when there is a broken build?
- How often do you consider adding automated processes to your build and CI system—on a continuous or even periodic basis?

But Tools are so important…

…Hudson/Jenkins is widely recognized as a **core tool for developers, QA engineers, project managers, release engineers, DevOps, and managers**. It is also a crucial open source option for teams using Agile methodology, in which continuous integration is a fundamental practice…

- *Kohsuke Kawaguchi - Creator of the Jenkins CI server. In:* 7 Ways to Optimize Jenkins/Hudson White Paper

The Hudson Book (pages 3-4)

- ***Without continuous integration, developing modern applications is practically impossible without dramatic inefficiency***. There is too much unproductive down-time, and far too much waste. The development group described above would be hard pressed to release anything more than a global update once a month
- ***In other words, when you automate build, test, and verify using a tool like Hudson you can continue developing your applications without having to wait (or synchronize) on some manual build, test, verify process.***

Just for information…

- ***Jenkins and Hudson are the same =) in***

http://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software

- **Why Hudson and Jenkins names were choosed?**

- ***Jenkins*** is an open source continuous integration tool written in Java. The project was ***forked*** from ***Hudson*** after a dispute with ***Oracle***, which claims the right to trademark the Hudson name and has applied for such a trademark as of December 2010

CI as a scheduling method

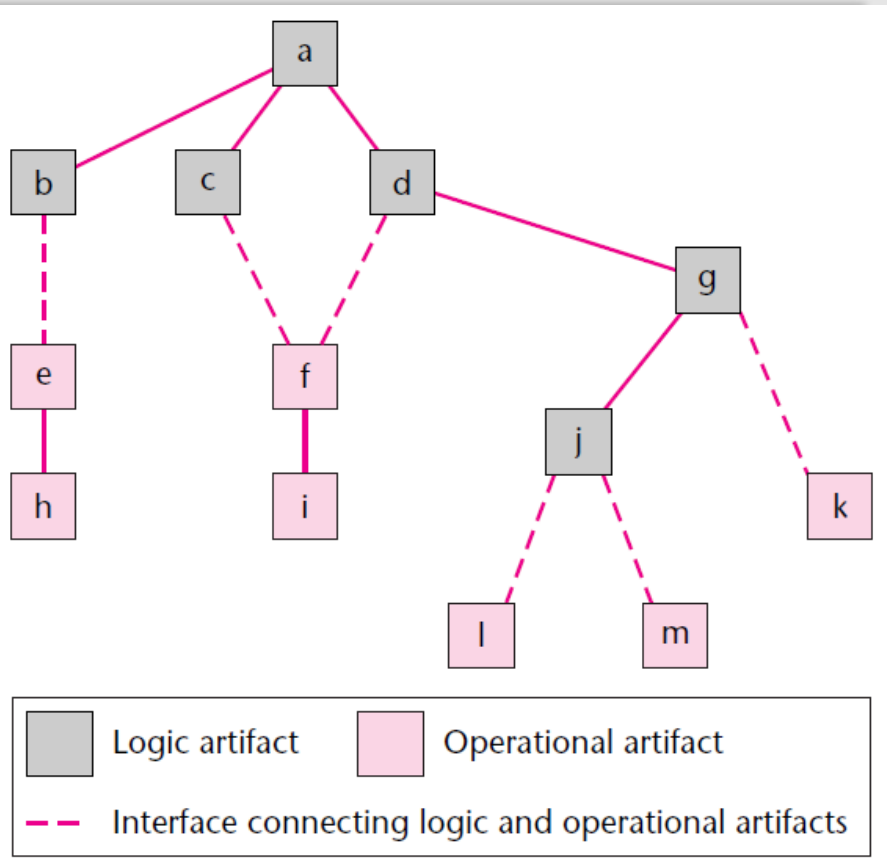
- ***RUP proposes the Integration Build Plan***
- The purpose of this is to **define the order in which the components should be implemented, which builds to create when integrating the system, and how they are to be assessed.**

CI as a scheduling method

- RUP: You should adjust the outline of the Integration Build Plan to **suit the nature of your project**. For example, if the system is large, there may be a case for having subsidiary plans for each implementation subsystem. The formality and extent of the test material contained or referenced from the Integration Build Plan will vary depending on the significance of the build.
- The RUP approach to integration is to ***incrementally integrate the software***. Incremental integration means that code is written and tested in small pieces, and then combined into a working whole by adding one piece at a time.

Integration approaches

- Integration then integration
- Top-down Integration
- Bottom-up integration
- Sandwich integration



Source: *Object-Oriented and Classical Software Engineering Eighth Edition*

Integration approaches

Approach	Strengths	Weaknesses
Implementation then integration	—	No fault isolation Major design faults show up late Potentially reusable code artifacts are not adequately tested
Top-down integration	Fault isolation Major design faults show up early	Potentially reusable code artifacts are not adequately tested
Bottom-up integration	Fault isolation Potentially reusable code artifacts are adequately tested	Major design faults show up late
Sandwich integration	Fault isolation Major design faults show up early Potentially reusable code artifacts are adequately tested	—

Source: *Object-Oriented and Classical Software Engineering Eighth Edition*

Integration approaches

How to Perform Sandwich Integration

- **In parallel,**
  Implement and integrate the logic artifacts top down.
  Implement and integrate the operational artifacts bottom up.
- Test the interfaces between the logic artifacts and the operational artifacts.

Source: *Object-Oriented and Classical Software Engineering Eighth Edition*

Thanks for your attention

fdgiraldo@uniquindio.edu.co

FACULTAD DE INGENIERÍA

UNIVERSIDAD DEL QUINDÍO