LT-QAI Torch Talk: 
Continuous Integration

Marc Schröder
<firstname.lastname>@dfki.de

DFKI Language Technology Lab, 
Saarbrücken and Berlin, Germany
What do I mean by Torch Talk?

- Idea to increase the frequency of LT-QAI talks on software engineering topics
  - encourage people to “dare” make a presentation
  - lower the entry barrier
  - presenter makes to claim to be an expert
  - some first experience with a technology / a workflow / some good practice
  - present, discuss, jointly decide whether to dig into the topic more deeply in some context
What do I mean by Torch Talk?

Why “Torch”? 
- a torch brings a bit of light into an unknown area 
- you can light a torch with little effort 
- depending on the others in the group, the torch might 
  - put something bigger “on fire”,
  - or be quietly extinguished without much wasted effort
Continuous Integration

Maybe the “core” of agile practices

- Scenario: team works together on a code base
  - different team members work on different parts
  - how do you integrate the parts?

- Old approach:
  - every individual or subteam develops their component in isolation
  - integration phase before release (painful)

- New approach:
  - Team integrates continuously!
Source code management

- Probably current practice in most of our projects today
- Everybody's code is “committed” to a repository, checked out to local “working copy”
  - Subversion, CVS, Mercury, Git

- Important element for Continuous Integration:
  - Whole team works on the same repository
  - People commit often (at least once per day)
    - but before I commit, my changes must reach a state where they don't break anything!
Single target build

Automate the build such that the full system can be built automatically, on a new machine, with no manual adaptation

- automatic resolution of any dependencies to third-party code
- automatic resolution of any dependencies between sub-projects
Continuous Integration Server

Basic idea:
- a server is running all the time, watching a source code repository for changes
- when a commit is detected, the latest code is automatically built on the CI server machine
- if the build breaks, the team can be notified
- if the build breaks, the team will make fixing the build its top priority
CI server

basic benefit:

- no more “it works on my machine”
- quick feedback when you forget to check in that new .java file
- establish trust that you know about the stability of the code
How to notify the team

• Through email
  ➡ to the whole team or to the person who broke the build
• Through instant messaging
• Through color of mobile phone status light
• Through lava lamps
  ➡ red and green lava lamp in team room
  ➡ idea: red lamp should never bubble
• …
Added benefits

Many tools have been developed in recent years which help identify problems in the code
- unit tests
- integration tests
- static code analysis
  - FindBugs for java
  - lint for C/C++
  - ...
- coverage measures
  - which percentage of methods is tested by my unit tests?
- finding duplicate code
  - the root of all evil, maintenance nightmare

All these can be run automatically on the CI server
Distinguish two tasks

**CI server**
- give **quick** feedback on compilation and unit tests
- has to run quickly (a few minutes max)
- has to run often (ideally after every commit)

**Integration**
- more extensive and time-consuming tasks can run once per night
  - run integration tests: does the entire system produce the expected output?
  - run static code analysis
  - build javadoc
  - build a nightly installable package
CI server software

- Idea is more than ten years old
- Now a good choice of easy-to-use free software is available
  - Apache Hudson / Jenkins
    - easy to configure via web interface
  - Apache Continuum
    - more fine-grained control
  - CruiseControl
    - was the first
    - configuration via XML files
Some hands-on experience with Hudson

- CI + nightly build for MARY TTS
- CI in Aliz-E for Java and C++