School of Computer Science and Engineering

INTRODUCTION TO DATA MANAGEMENT

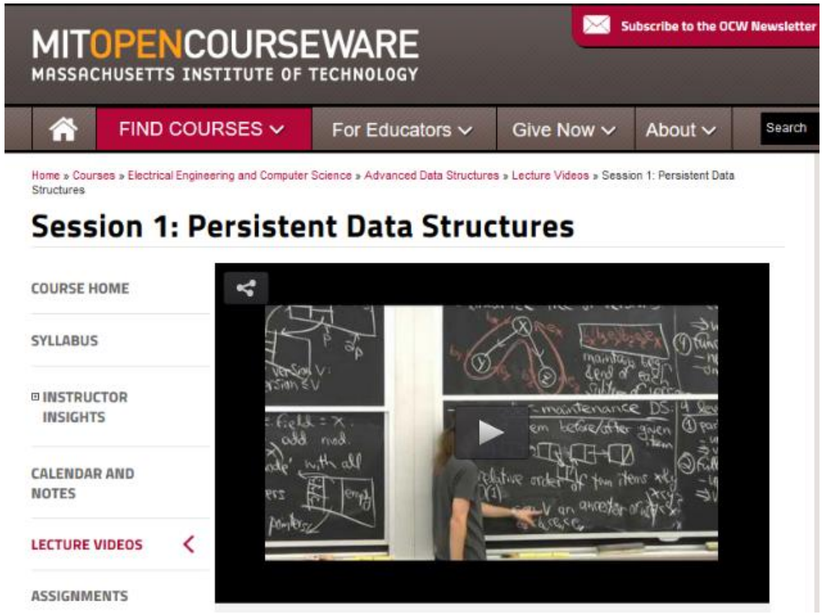
Lecture 5

Persistent Structures. Continuous Integration

Evgeny Pyshkin
Senior Associate Professor
Software Engineering Lab

Maxim Mozgovoy
Senior Associate Professor
Active Knowledge Lab

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-851-advanced-data-structures-spring-2012/lecture-videos/session-1-persistent-data-structures/



Home » Courses » Electrical Engineering and Computer Science » Advanced Data Structures » Lecture Videos » Session 1: Persistent Data Structures

Session 1: Persistent Data Structures

Unit 1

UNDERSTANDING PERSISTENCE

Let’s Watch a Movie

https://www.youtube.com/watch?v=mnJegNyAb1w

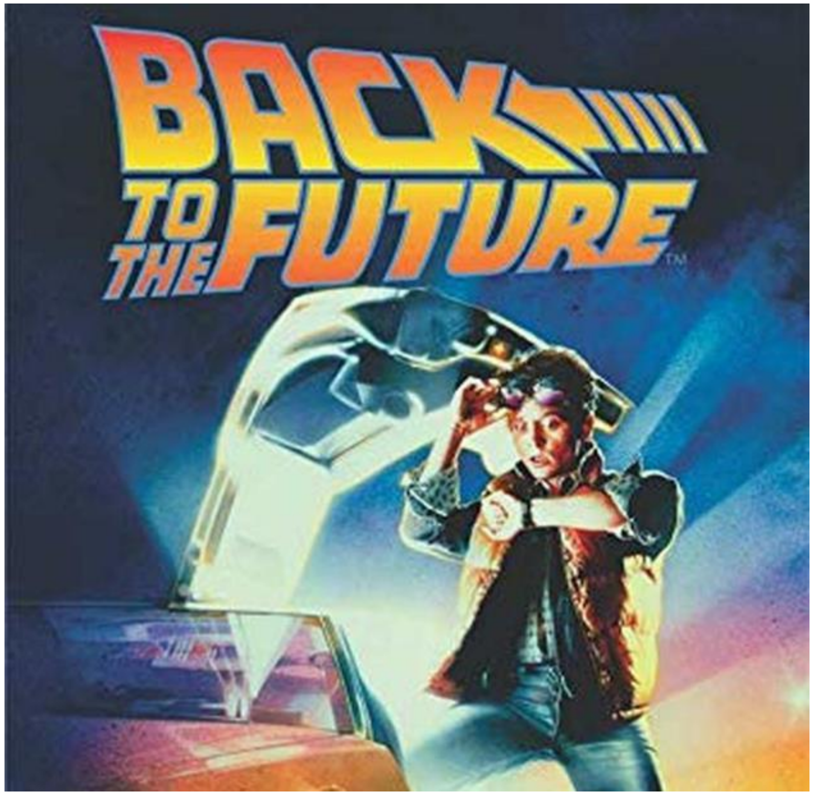
PLEASE WATCH AT YOUR DEVICE!

https://www.youtube.com/watch?v=UTO0ogdNMdY

HIMPFF 2017
Roswell Film Festival
LAIFFA Official Selection 2017
Los Angeles CineFest 2017
Hollywood Screenings Film Festival 2017
CANADASHORTS
0:00 / 26:00

https://www.youtube.com/watch?v=ksSYPpV9bdc

An important concept ☺ at 1’49 (resource management)



https://www.youtube.com/watch?v=vBkBS4O3yvY

An important concept ☺ at 3’35 (making copies)

Time Travel

- According to Eric Demaine (MIT), there are two senses in time travel (or **temporal data structures**)
  - Persistence
    - “Branching universe time travel model”
      - If we make a change in the past, we create a new universe
      - We never destroy old universes
      - We do not forget anything
  - Retroactivity
    - “Back to the Future”
      - We go back, make a change, return to the present and see what happened
      - It is more complicated thing compare to persistence

Persistent Data Structures

*

- A **persistent data structure** is a data structure that always preserves the previous version of itself
  - The general idea is to keep all the versions of data structures
  - Data structure operations are all relative to a specific version of data structure
  - Why keep all the versions?
  - Persistence in software development
    - Persistent structure are effectively immutable, as their operations do not (visibly) update the structure in-place, but instead always yield a new updated structure

** Kazimir Malevich, Black Square, 1915, Tretyakov Gallery, Moscow.*

Understanding Immutability

- Context
  - Objects that share references to the same object => possible improper coordination between the objects sharing another object
  - Assuring correct object coordination requires careful programming. There are error risks.
  - Particular case – concurrent threads using shared objects

```
Date d = new Date();
Scheduler.scheduleTask(task1, d);
d.setTime(d.getTime() + ONE_DAY);
scheduler.scheduleTask(task2, d);
```

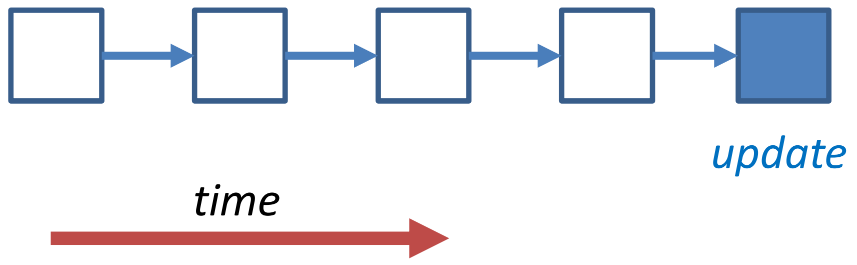
- Solution
  - Organizing a class so that the state information of its instances never changes after they are constructed
  - No method, other than a constructor, should modify the values of a class's instance variables
  - Change operations are disallowed
  - In operation that might change object new instance is created instead
  - *Result: there is no shared changeable objects*

Four Levels of Persistence: Outline

- Partially persistent structure
  - Only the latest version update is allowed.
  - We can access any previous version.
- Fully persistent structure
  - We can update any version.
  - Queries are allowed for any version as well.
- Confluent persistent structure
  - We can combine two or more versions.
- Functional persistent structure
  - We can never modify anything
  - Only producing new nodes is allowed.

Partial Persistence

- Update the latest version -> versions are linearly ordered.
- It allows looking back at the past to the older version versions, but you can change only the most recent one. That’s like simple time machine.



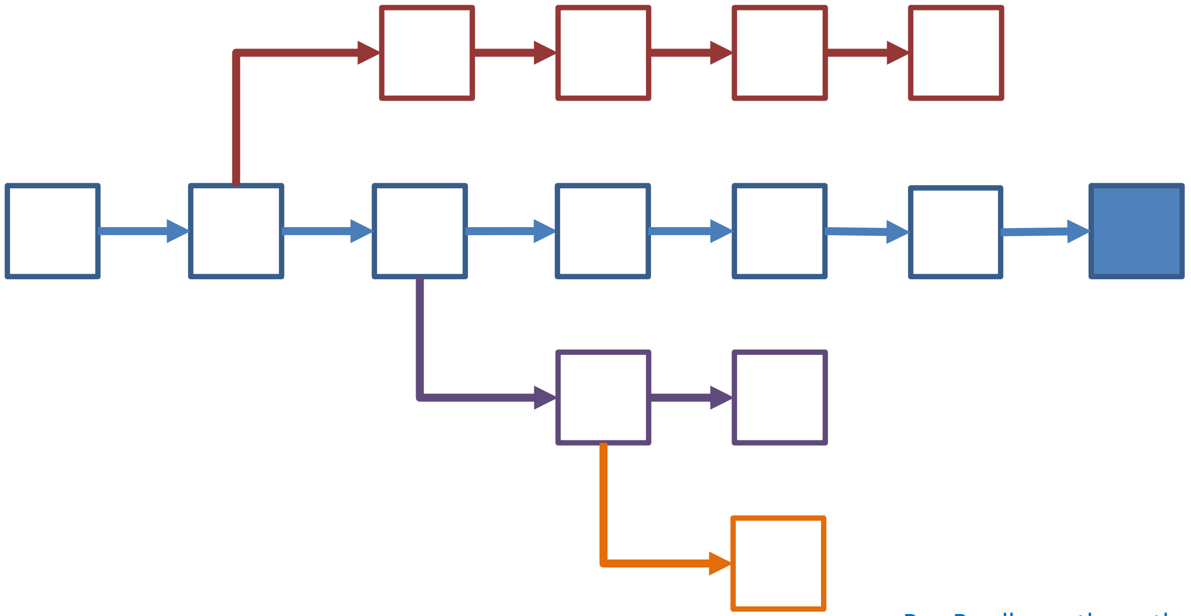
Theoretically,
Constant time
for
Both adding new
nodes and mods

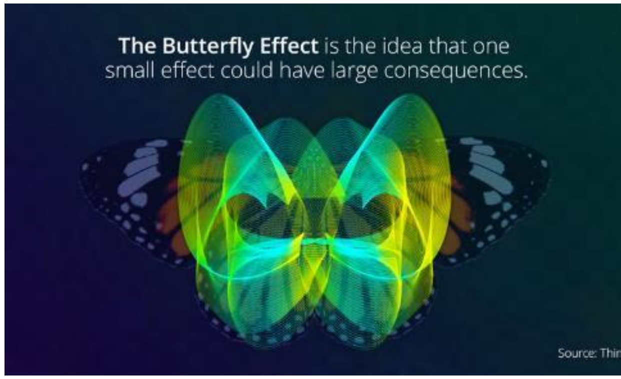
- DejaVu movie (not very common but good time travel movie):
  - In the first half all that the heroes do is to look at the past, but later they discover that actually they have a…

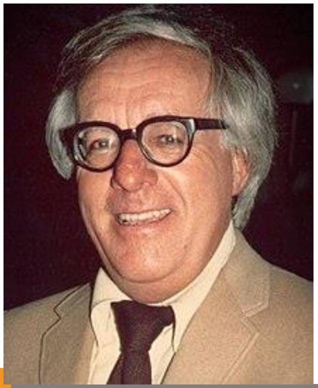
Full Persistence

- You can update anything you want
- => versions are not linearly ordered -> they form the tree. “Branching universe model”.
- Many problems, so we need the third level of persistence.

“Butterfly Effect” single cover, Shiritsu Ebisu Chugaku

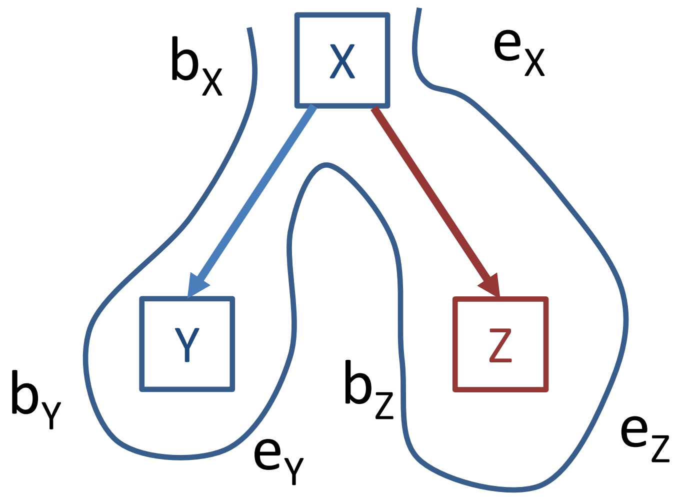



The Butterfly Effect is the idea that one small effect could have large consequences.

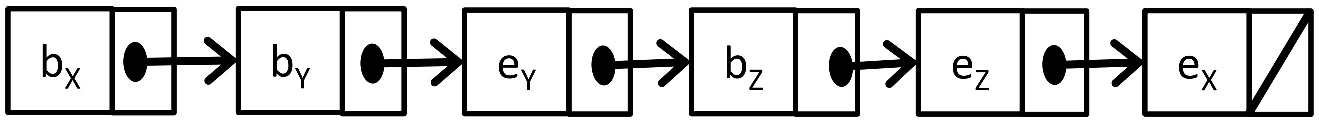

Ray Bradbury, the author of the “Sound of Thunder” (1952)

Full Persistence

- We can linearize the tree of versions to maintain begins and ends of each subtree of versions

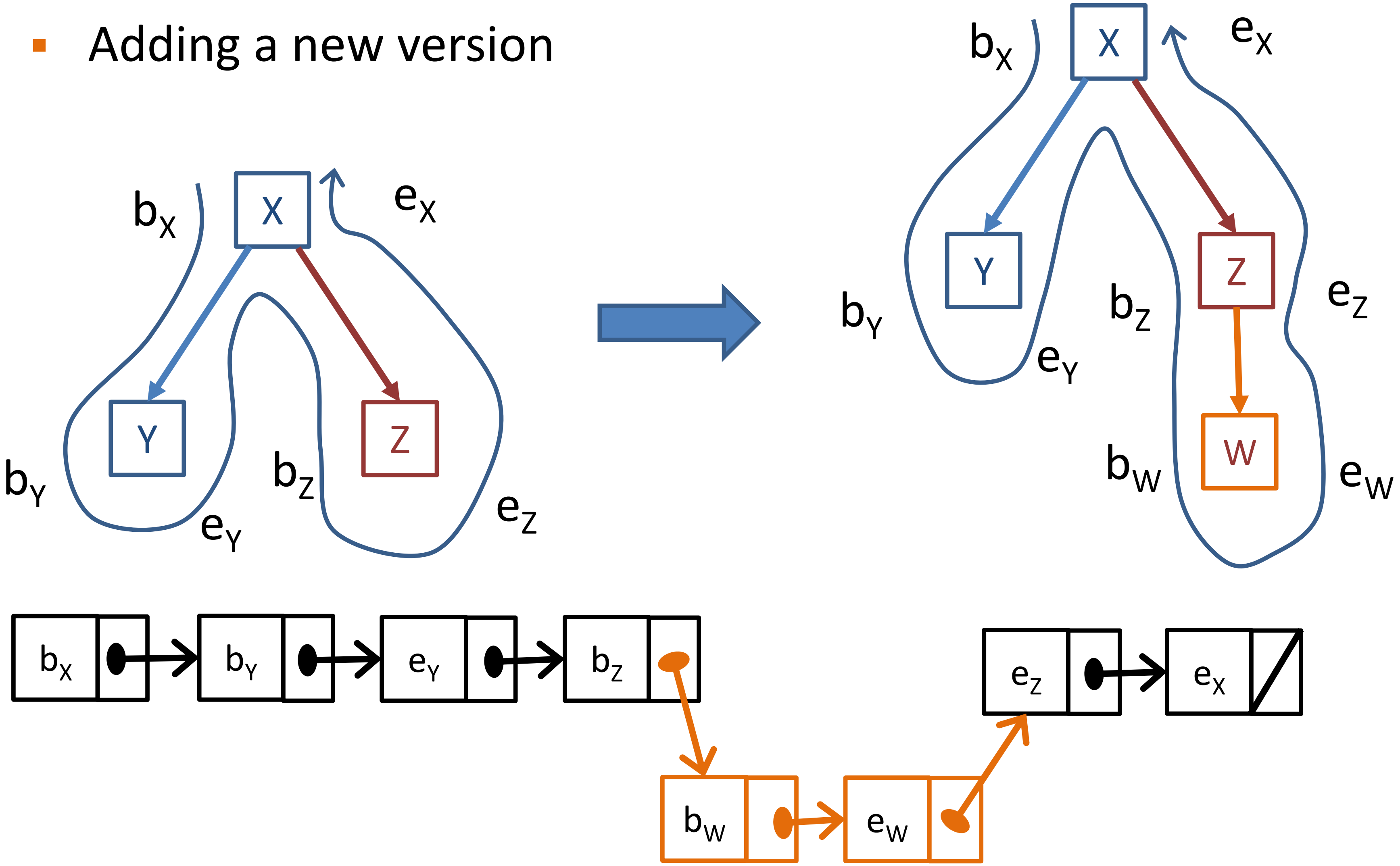


Theoretically,
Constant time
for
Both adding new
nodes and mods



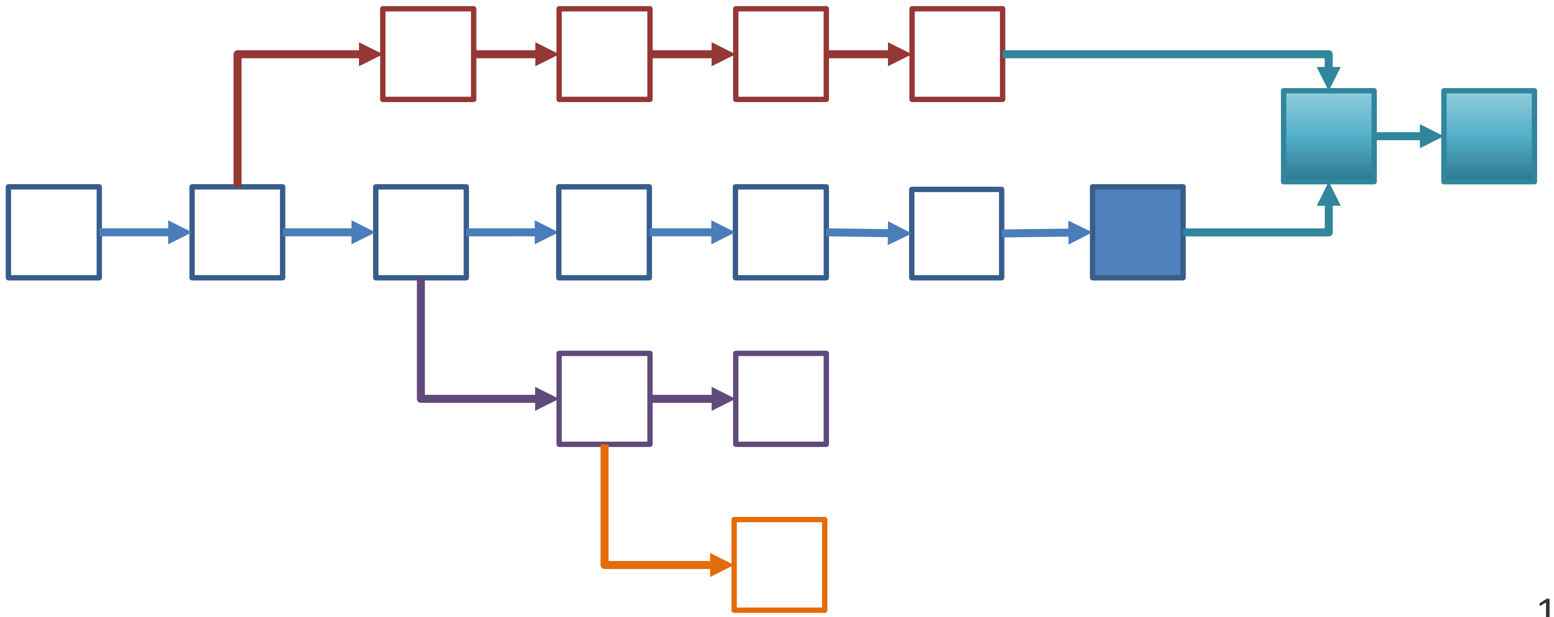
Full Persistence

- Adding a new version



Confluent Persistence

- You can merge versions, every merge creates a new version.
- The versions form the directed acyclic graph (DAG).
- Very hard to do!

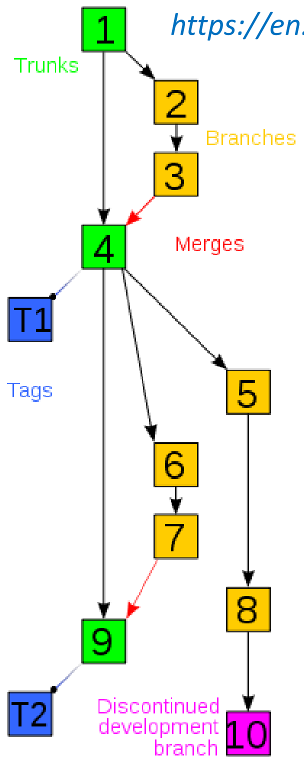


Functional Persistence

- We can never modify anything.
  - An update is not a modification.
  - All you can do is to create new nodes.
- In a purely functional software model all data are immutable, so all data structures are automatically fully persistent.

Some more home reading
James R. Driscoll et al. “Making Data Structures Persistent” (1986):
https://www.cs.cmu.edu/~sleator/papers/another-persistence.pdf

Eric D. Demaine et al. “Retroactive Data Structures” (2007):
https://dl.acm.org/citation.cfm?doid=1240233.1240236



https://en.wikipedia.org/wiki/Version_control

Unit 2

PERSISTENCE IN SOFTWARE

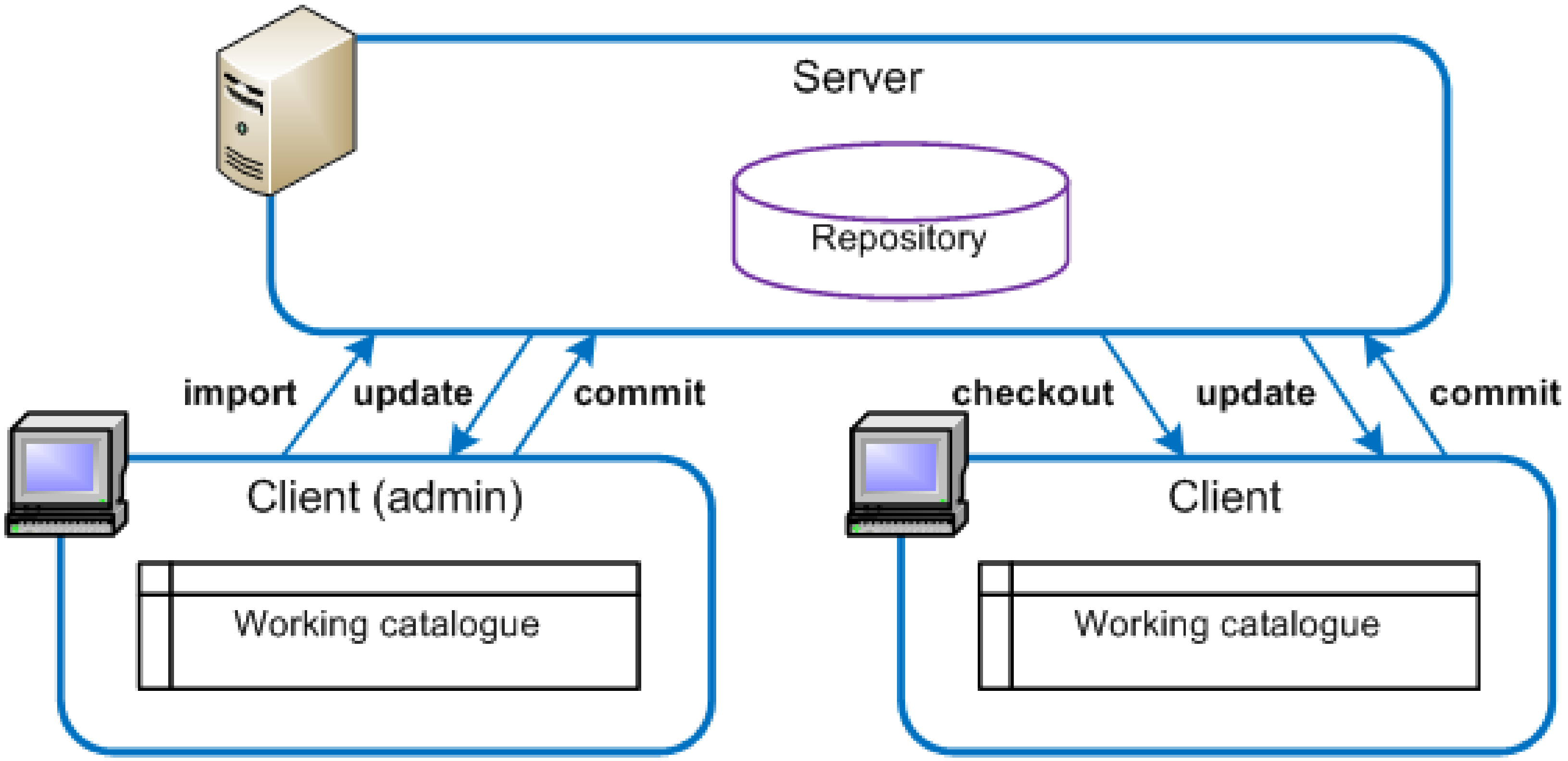
Why Versions?

- Accessing files by different team members
  - “Simultaneous” editing
- Different project revisions
  - Release for customers
  - Version under development
- Different configurations
  - To support multiple operating systems/platforms
  - To support different hardware
- Distributed development
- Open source development

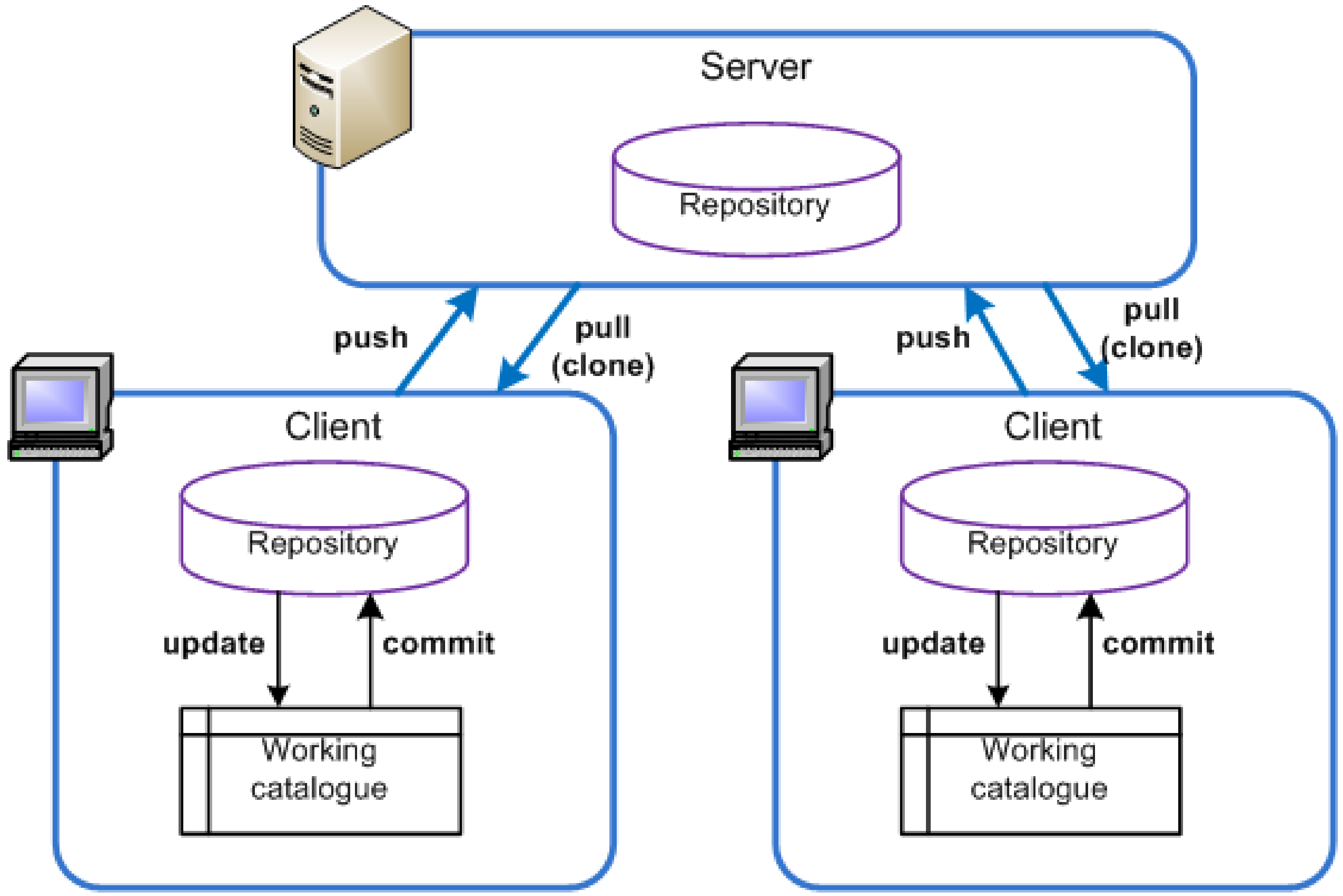
Version Control Systems

- Version control systems for software developers. Motivation and three generations of VCSs. Typical operations and workflow. The problem of binary data versioning.
- Repositories. Repository operations. Approaches: centralized (SVN) and distributed (Mercurial).
- Will be discussed in details in Lecture 6.

Centralized Model (Subversion, CVS, etc.)



Distributed Model (Mercurial, git, etc.)



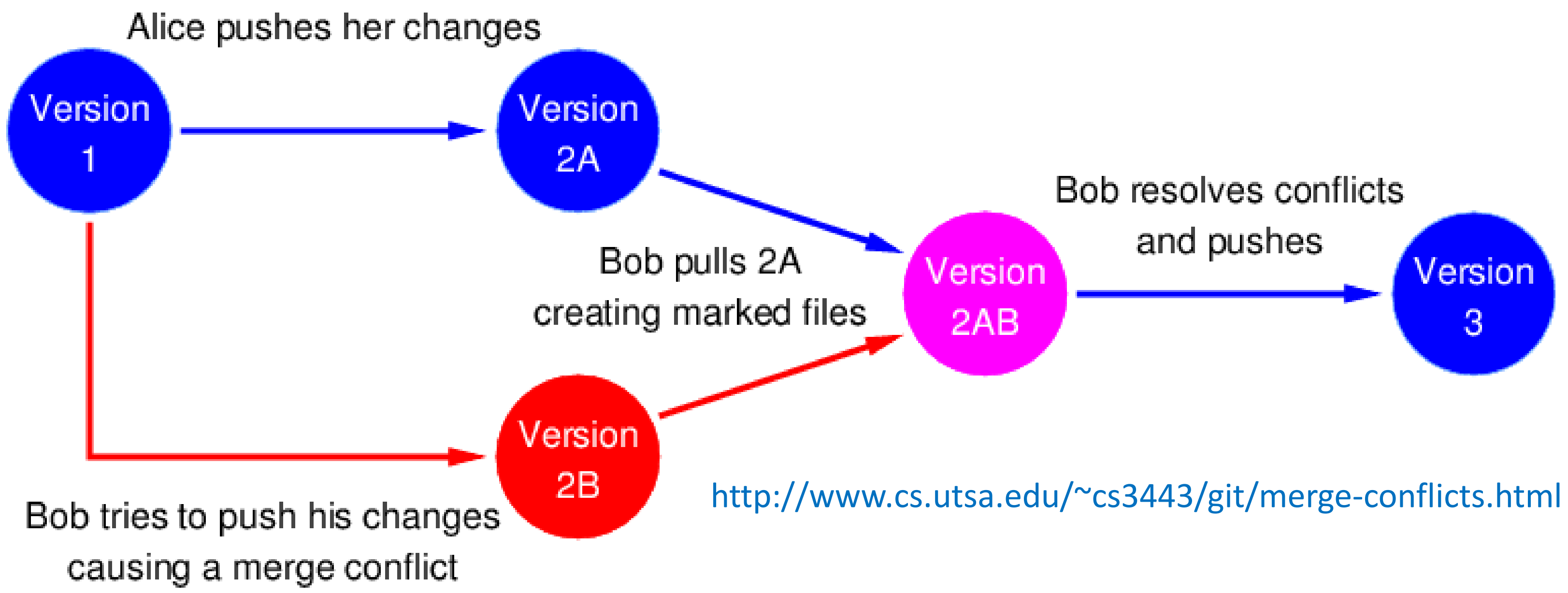
Concept of Atomic Operation

- An operation is *atomic* if the system is left in a consistent state even if the operation is interrupted. The *commit* operation is usually the most critical in this sense. Commits tell the revision control system to make a group of changes final, and available to all users.
  - Not all revision control systems have atomic commits; notably, CVS lacks this feature.

Common Version Control Operations

- Import
  - Initial allocation of the project on the version control system repository.
- Check in
  - Getting the copy of a certain version to the local repository.
- Check out (update)
  - Copies the latest version of the requested files from the VCS repository and merges the local files with the update (if necessary).
- Commit
  - Copies changes in the local file system to the version control repository.
- Lock
  - Prevents "concurrent access“ o the file: only one developer at a time has write access to the central "repository" copies of those files. Once one developer "checks out" a file, others can read that file, but no one else may change that file until that developer "checks in" the updated version (or cancels the checkout).
- Diff
  - Displays a tool with the differences between files on the local file system and files in the repository.
- Log
  - Displays a list of all the revisions, with commit comments, for the files that are in the scope of the command.
- Add
  - Requests that a file or directory be added to the next commit.
- Remove
  - Requests that a file or directory be removed in the next commit.
- Revert
  - Disposes of any local changes and reverts the local files to match the current revision in the repository.

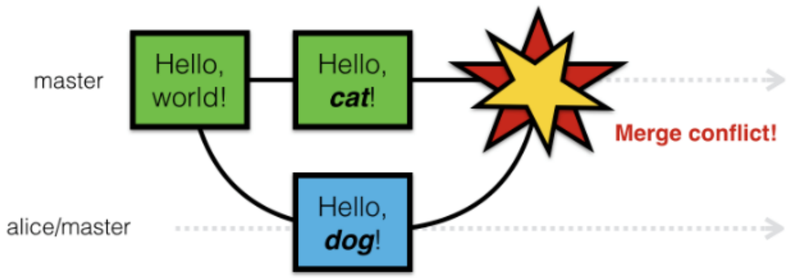
Resolving Merge Conflicts in Commits


Alice pushes her changes
Version 1
Version 2A
Bob pulls 2A creating marked files
Version 2AB
Bob resolves conflicts and pushes
Version 3
Version 2B
Bob tries to push his changes causing a merge conflict

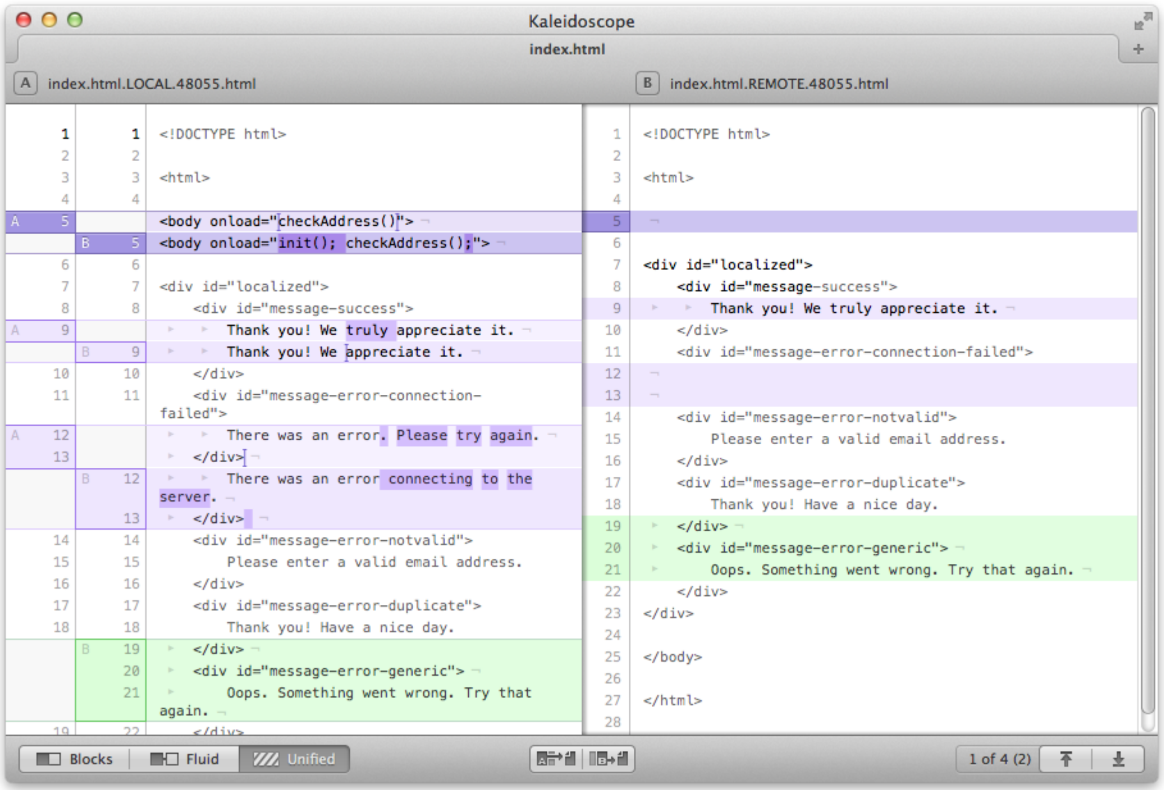
http://www.cs.utsa.edu/~cs3443/git/merge-conflicts.html

- The most common situations:
  - Competing changes are made to the same line of a file;
  - A file is deleted that another person is attempting to edit.
- Useful tips:
  - Build your commit and run automated teste before commit
  - Communicate with your co-workers
  - If you are unsure of a merge, do not force it.

Comparing Differences



- Version Control Systems provide tools to compare differences (see the next lecture!)

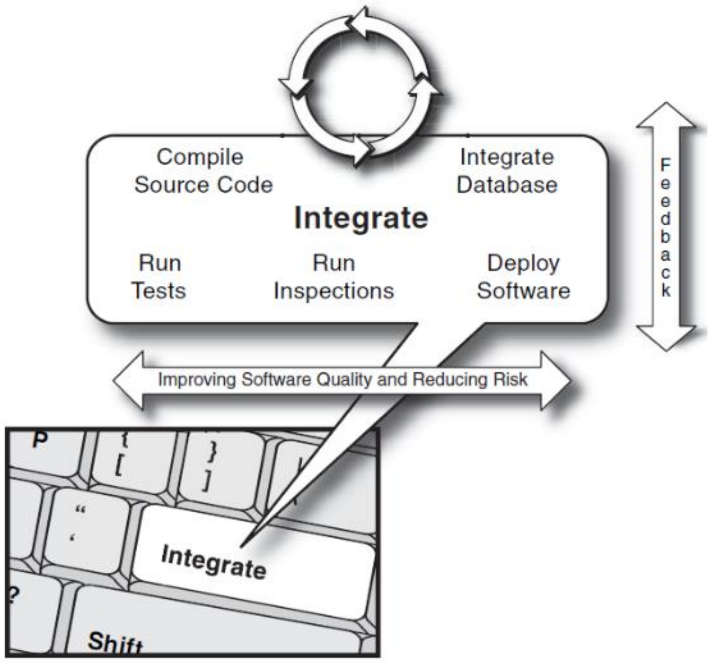


```
<!DOCTYPE html>

<html>

<body onload="checkAddress()">
<body onload="init(); checkAddress();">

<div id="localized">
    <div id="message-success">
        Thank you! We truly appreciate it.
        Thank you! We appreciate it.
    </div>
    <div id="message-error-connection-failed">
        There was an error. Please try again.
    </div>
        There was an error connecting to the server.
    </div>
    <div id="message-error-notvalid">
        Please enter a valid email address.
    </div>
    <div id="message-error-duplicate">
        Thank you! Have a nice day.
    </div>
    <div id="message-error-generic">
        Oops. Something went wrong. Try that again.
    </div>
```

```
<!DOCTYPE html>

<html>



<div id="localized">
    <div id="message-success">
        Thank you! We truly appreciate it.
    </div>
    <div id="message-error-connection-failed">


    <div id="message-error-notvalid">
        Please enter a valid email address.
    </div>
    <div id="message-error-duplicate">
        Thank you! Have a nice day.
    </div>
    <div id="message-error-generic">
        Oops. Something went wrong. Try that again.
    </div>
</div>

</body>

</html>

```

https://stackoverflow.com/questions/9207260/merge-conflict-resolution

Paul Duval, Steve Matyas, and Andrew Gloves, "Continuous Integration. Improving Software Quality and Reducing Risk," Pearson Education, Inc., 2007.

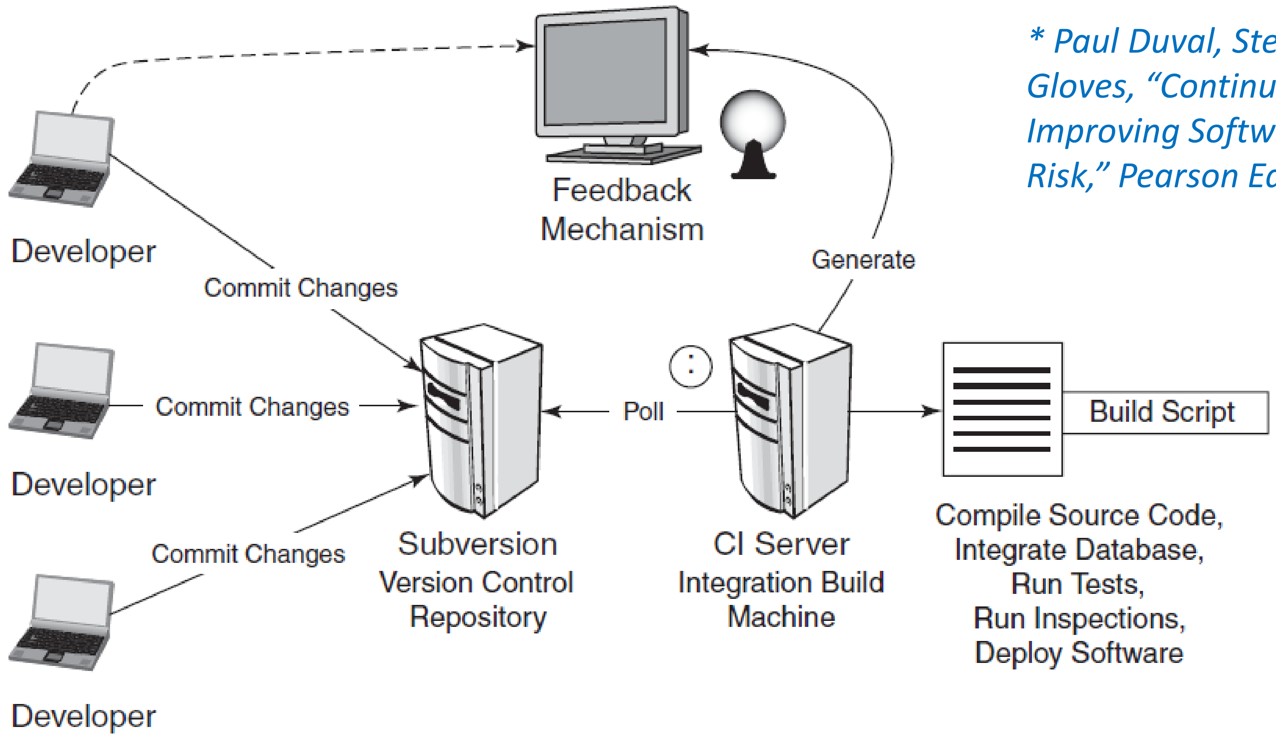


Unit 3

CONTINUOUS INTEGRATION

The Components of a CI System*

- “Once a developer has performed all of the modifications related to the assigned task, she runs a private build (which integrates changes from the rest of the team) and then commits her changes to the version control repository”*.



Feedback Mechanism
Developer
Commit Changes
Developer
Commit Changes
Commit Changes
Developer
Generate
Poll
Subversion Version Control Repository
CI Server Integration Build Machine
Build Script
Compile Source Code, Integrate Database, Run Tests, Run Inspections, Deploy Software

** Paul Duval, Steve Matyas, and Andrew Gloves, “Continuous Integration. Improving Software Quality and Reducing Risk,” Pearson Education, Inc., 2007.*

What is the Value of CI

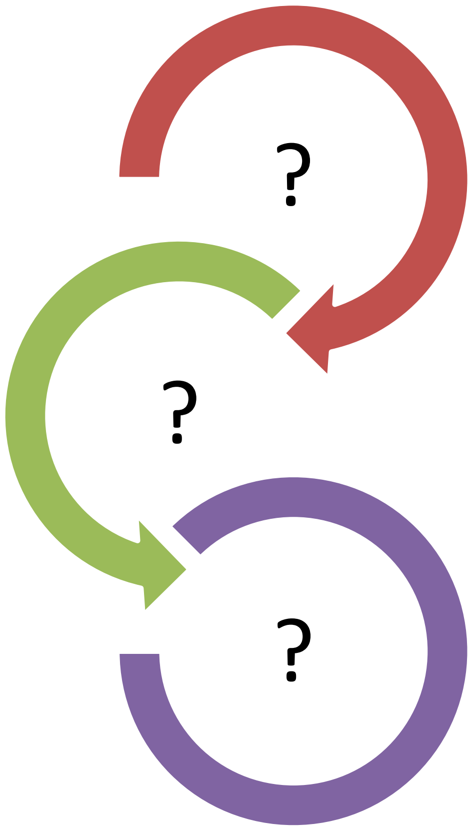
- Reduce risks
  - Defects are detected and fixed sooner
  - Health of software is measurable
- Reduce repetitive manual processes
  - The process runs the same way every time
  - The processes will run every time a commit occurs in the version control repository
- Generate deployable software at any time and at any place
  - You make small changes to the source code and integrate these changes with the rest of the code base on a regular basis
  - If there are any problems, the project members are informed and the fixes are applied to the software immediately
- Enable better project visibility
  - A CI system can provide just-in-time information on the recent build status and quality metrics
- Establish greater confidence in the software product from the development team
  - With every build, your team knows that tests are run against the software to verify behavior

Simple (?) Practices

- If you are individual or a team running CI on a project:
  - Commit code frequently
  - Don’t commit broken code
  - Fix broken builds immediately
  - Write automated developer tests
  - All tests and inspections must pass
  - Run private builds
  - Avoid getting broken code

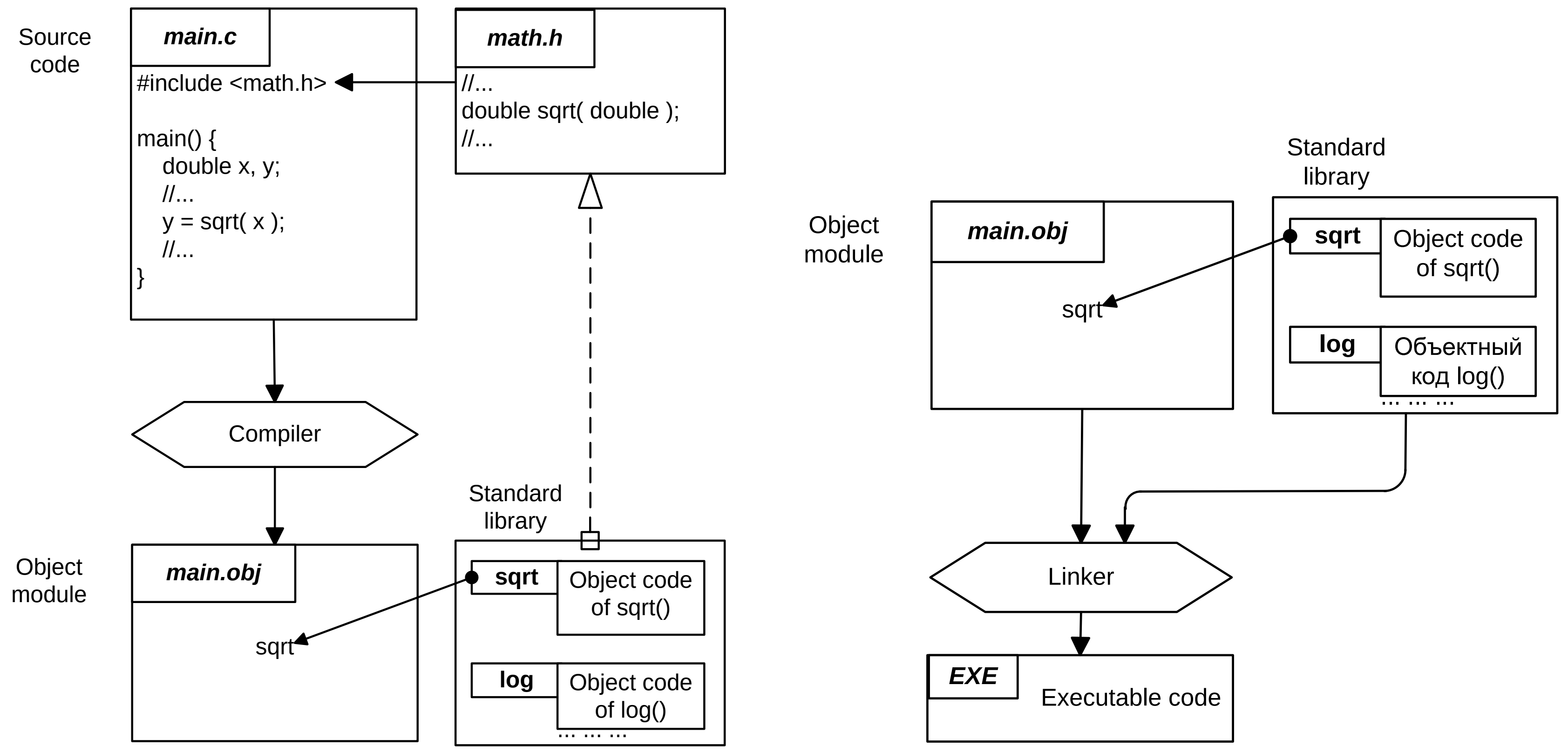
Build: Why is it Complex?

- Build: What does it mean?



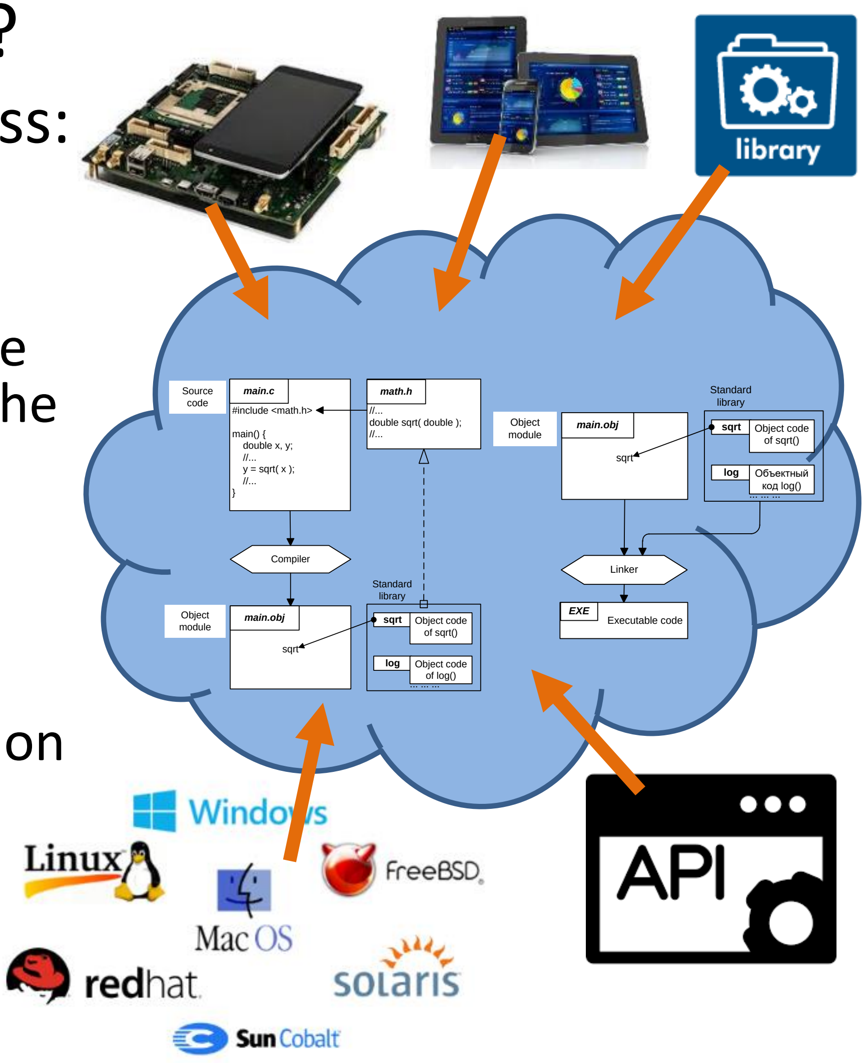
Build: Why is it Complex?

- Build: What does it mean?
  - From “C Programming” class: Compile + Link



Build: Why is it Complex?

- Build: What does it mean?
  - From “C Programming” class: Compile + Link
  - But it is more:
    - Automatic compilation of the whole project according to the recent changes
    - Linking all the modules
    - Archiving
    - Generating the source code
    - Generating the documentation
    - Deploying the project
    - Sending feedback to the subscribed users
    - …



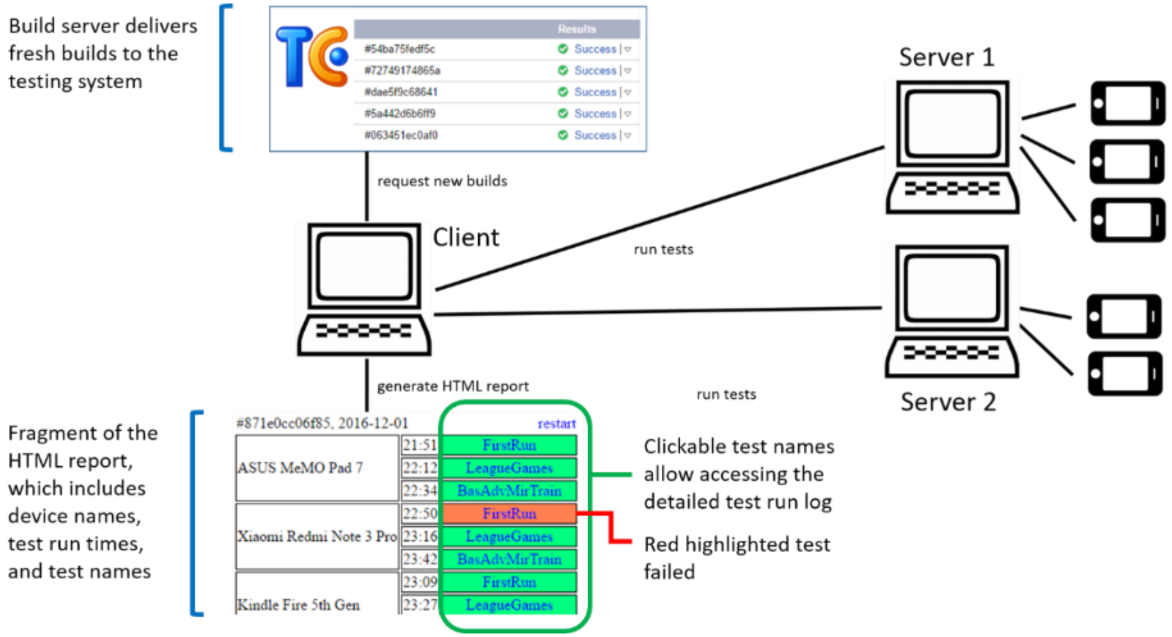
Build Artifacts and Tools

- Source codes
  - Probably written in different languages
- Libraries
  - Standard libraries
  - Third-party libraries
- Templates
- Configuration files
- Resources
- Build procedure definition
- Documentation
  - docbook, html, …

- Tools supporting automated builds:
  - TeamCity
  - IBM Rational Build Force
  - Visual Build
  - OpenMake
  - …

CI Is About Automation

- Automated workflow
  - Automated builds
  - Automated testing
  - Automated reports
  - …


Build server delivers fresh builds to the testing system
request new builds
Client
run tests
Server 1
run tests
Server 2
generate HTML report
Fragment of the HTML report, which includes device names, test run times, and test names
Clickable test names allow accessing the detailed test run log
Red highlighted test failed

	Results
#54ba75fedf5c	Success
#72749174865a	Success
#dae5f9c68641	Success
#5a442d6b6ff9	Success
#063451ec0af0	Success

#871e0cc06f85, 2016-12-01 restart

ASUS MeMO Pad 7	21:51	FirstRun
	22:12	LeagueGames
	22:34	BasAdvMirTrain
Xiaomi Redmi Note 3 Pro	22:50	FirstRun
	23:16	LeagueGames
	23:42	BasAdvMirTrain
Kindle Fire 5th Gen	23:09	FirstRun
	23:27	LeagueGames

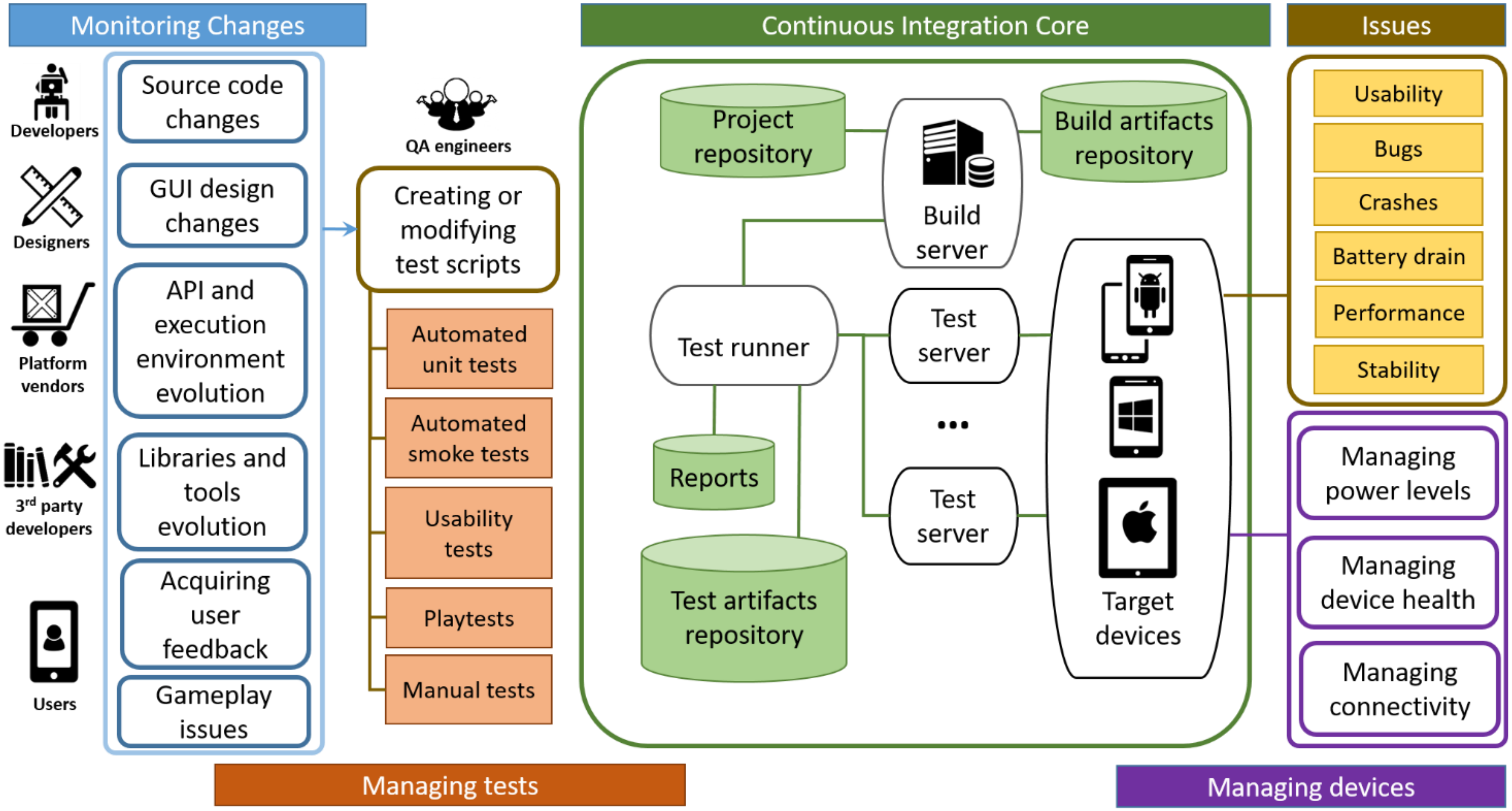
Maxim Mozgovoy and Evgeny Pyshkin, “Mobile Farm for Software Testing,” MobileHCI-2018.

Some more home reading

Paul Duval, Steve Matyas, and Andrew Gloves, “Continuous Integration. Improving Software Quality and Reducing Risk,” Pearson Education, Inc., 2007.

Martin Fowler, “Continuous Integration” (2006): https://www.martinfowler.com/articles/continuousIntegration.html

Example: Mobile Software Quality Assurance



Maxim Mozgovoy and Evgeny Pyshkin, "A Comprehensive Approach to Quality Assurance in a Mobile Game Project," CEE-SECR'18.

School of Computer Science and Engineering

INTRODUCTION TO DATA MANAGEMENT

Lecture 5

Persistent Structures. Continuous Integration

Q & (maybe) A