Confessions of a retired build manager

How one academic institution produces quality software with mixed crew of programmers and low overhead

Kate Wilczek
Human Genome Sequencing Center
Baylor College of Medicine

Content

- Software development challenges
- What HGSC does
- Types of software we write
- Software team composition and organization
- Build manager responsibilities
- Build tools as solutions to challenges
- One-step tag and build script

Software development challenges

- Unversioned artifacts in production
- Build breaks after dependency is updated
- IDE-dependent builds that can’t migrate into other IDEs or platforms
- Code is readable only on one IDE
- Chaos in code repository
- Missing dependencies
- Missing code documentation

What does HGSC do?

- One of three major genome sequencing centers in the country
- Finds causes of inherited human disorders
- Helps develop personalized cancer therapies
- Investigates influence of microbiota on human health
- Looks at genomes of cows, primates, armadillos, and even bugs

Software we write

- Data processing pipelines
- Client-server web applications
- HGSC website

We also maintain…

- Submissions of information to government and other public entities
- Databases
- Web server configurations

Software team

- Diversity of computer science and biology/ bioinformatics backgrounds
- One programmer handles more than one project
- GEN2LIMS is written by a team
- Programmers interact directly with users

We are multilingual

```
/**
 * class that contains methods to process BAM files
 * User: kylec
 * Date: Mar 24, 2010
 * Time: 6:42:49 PM
 * To change this template use File | Settings | File Templates.
 */
import net.sf.samtools.*;
import net.sf.samtools.util.CloseableIterator;

import java.io.File;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Hashtable;
import java.util.List;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.BitSet;
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class Bam {

    public static void main (String [] args) {

        if(args.length < 4) {
            System.out.println("usage: bam option(r=rename, t=target -c = clean -csbc=clean_and_split_by_chromosome)
f stringency");
            System.exit(0);

        }

        String option = args[0];
        String inputPath = args[1];
        String outputPath = args[2];
        String targetPath = args[3];
        int indel_cnt_cut= Integer.parseInt(args[4]);
        int var_cnt_cut= Integer.parseInt(args[5]);
        String validationStringency = args[6];

        try {
            if (option.equals("s")) {
                Bam.splitBamByRun(inputPath, outputPath);
            } else if(option.equals("t")) {
                System.out.println("make on-target Bam: "+ inputPath + "," + outputPath + "," + targetPath);
                Bam.getTargetBam(inputPath, outputPath, targetPath, validationStringency);
```

We are multilingual

```
#!/usr/bin/env groovy

import groovy.io.*
import java.io.File
import groovy.sql.Sql

if (args.size() == 0) {
    println "no data source provided.  Terminating..."
} else {
    def fileName = args[0]
    def inputFile = new File(fileName)
    inputFile.splitEachLine('\t') {
    fields ->
        def project_name = fields[0]
        def description = fields[1]
        def sampleNumber = fields[2].toInteger()

    }


    def sqlOracle = Sql.newInstance("jdbc:oracle:thin:@hondo.hgsc.bcm.tmc.edu:1521:gscdevel", "mutations",
"mutations",    "oracle.jdbc.driver.OracleDriver")
```

We are multilingual

```perl
#!/usr/bin/env perl
#finds all contigs whose actual length is less than a given number of nucleotides
#USAGE: ./findAceCntigLengh.pl file-name
#AUTHOR: Kate Wilczek
#DATE: 2010-12-16

$file_name = $ARGV[0];
$ace_file_name = $file_name.".ace";
print "$ace_file_name\n";
$validator_output = $file_name.".val";
open FILE, "< $ace_file_name" or die "Cannot find file: $!";
open VALIDATIONFILE, ">$validator_output" or die "Cannot open file for writing: $!";
$lineindex = 0;
$min_contig_length = 200;
while ($next_line = <FILE>) {
    $lineindex += 1;
    if ($next_line =~ m/^CO\s(\S+)\s(\d+)\s(\d+)\s(\d+)\s(\w+)\n/) {
        @contig_header = split(/\s/, $next_line);
        $contig_name = @contig_header[1];
        $sequence_trimmedLengh = 0;
        $sequence_line = <FILE>;
        chomp ($sequence_line);
        $contig_sequence =+ $sequence_line;
        while ($sequence_line =~ m/(\w+)/){
            $sequence_line = <FILE>;
            chomp ($sequence_line);
            $contig_sequence = $contig_sequence . $sequence_line;
        }
        $sequence_initialLength = length ($contig_sequence);
        $initialPosition = 0;

        while ($initialPosition < $sequence_initialLength) {
            $nucleotide = substr ($contig_sequence, $initialPosition, 1);
            if ($nucleotide =~ m/[A-Za-z]/) {
                $sequence_trimmedLength += 1;
            }
            $initialPosition += 1 ;

        }
        if ($sequence_trimmedLength < $min_contig_length){
            $error_message = "ERROR: $contig_name contains only $sequence_trimmedLength nucleotides\n";
            print VALIDATIONFILE $error_message;
        }
        $sequence_trimmedLength = 0;
    }
}
close FILE;
close VALIDATIONFILE;
```

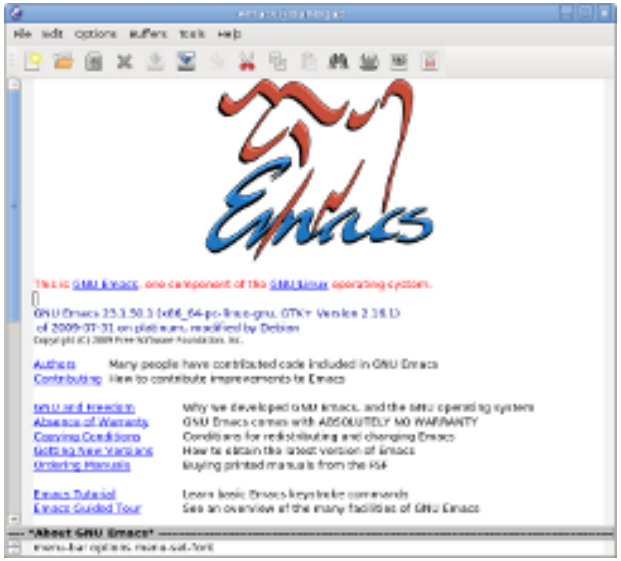
We are multilingual

```
<!--
    Area below the title panel that displays the background color panel.
    It encloses both the colorPanel div and the content div.
-->
<xsl:element name="div">
     <xsl:attribute name="id">
        <xsl:choose>
                <xsl:when test="//content/frame[@display = 'wide'] or $menu = 'off'">enclose4nobackground</xsl:when>
                <xsl:when test="imagedata/@background = 'none'">enclose-noBackground</xsl:when>
                <!-- <xsl:when test="imagedata/@background = 'DNAhelix'">enclose-DNAhelixBackground</xsl:when> -->
                <!-- <xsl:when test="imagedata/@background = 'long'">enclose-long</xsl:when> -->
                <xsl:when test="$browser = 'explorer' ">enclose4IE</xsl:when>
                <xsl:otherwise>enclose4</xsl:otherwise>
        </xsl:choose>
    </xsl:attribute>

    <!--
        MENU

    <xsl:if test="not(menuoff)">
        -->
     <xsl:if test="not($menu = 'off')">
    <!--  <xsl:if test="not(//content/frame[@menu = 'off'])">-->
        <div>
            <xsl:attribute name="id">
            <xsl:choose>
                <xsl:when test="$browser = 'explorer'">navMenuSpeciesIE</xsl:when>
                <xsl:otherwise>navMenuSpecies</xsl:otherwise>
            </xsl:choose>
            </xsl:attribute>
              <ul class="navMenuButtons">
                <xsl:apply-templates select="//levels[@page='redesign']"/>
              </ul>
        </div>
    </xsl:if>
```

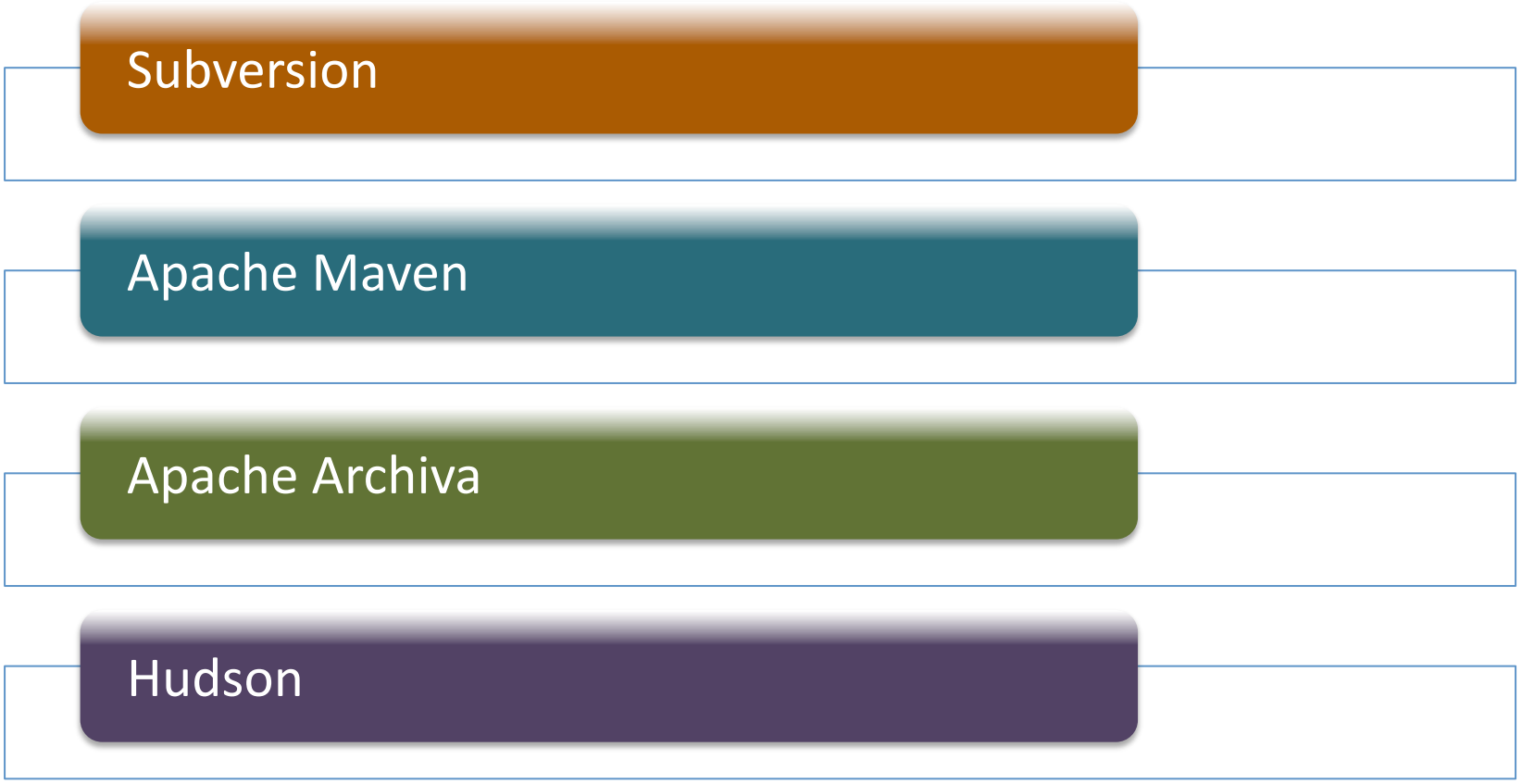
We are multiplatform



Build manager tasks

- Builds and tags individual version releases
- Decides on new version number
- Reviews code with individual programmers
- Deploys external libraries

Build tools as solutions



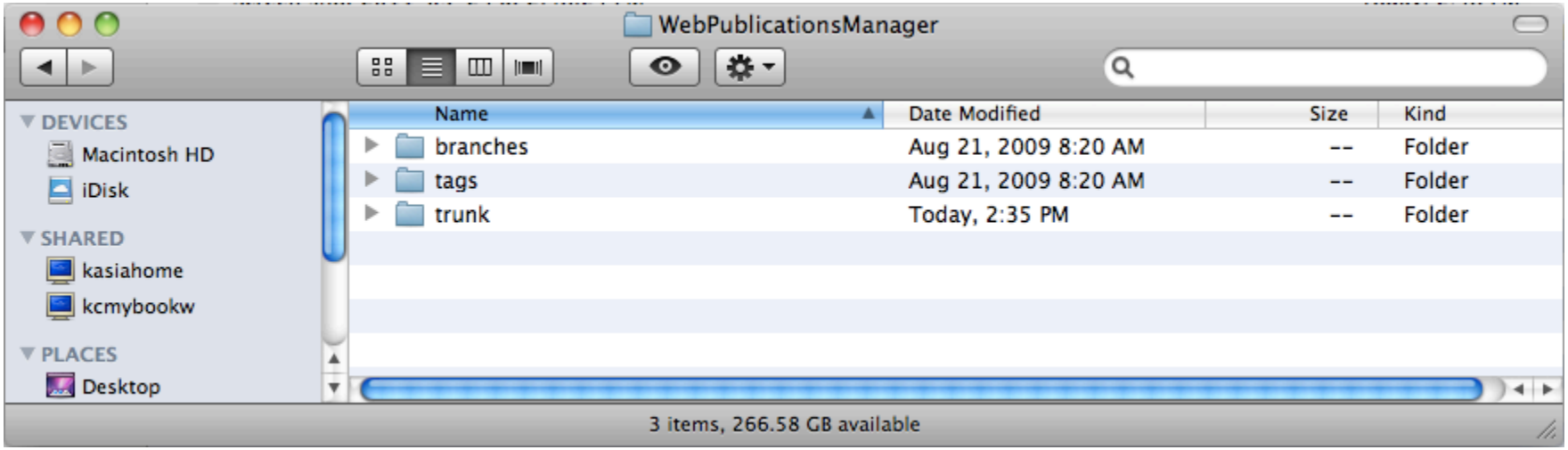
Solution to repository chaos

Subversion

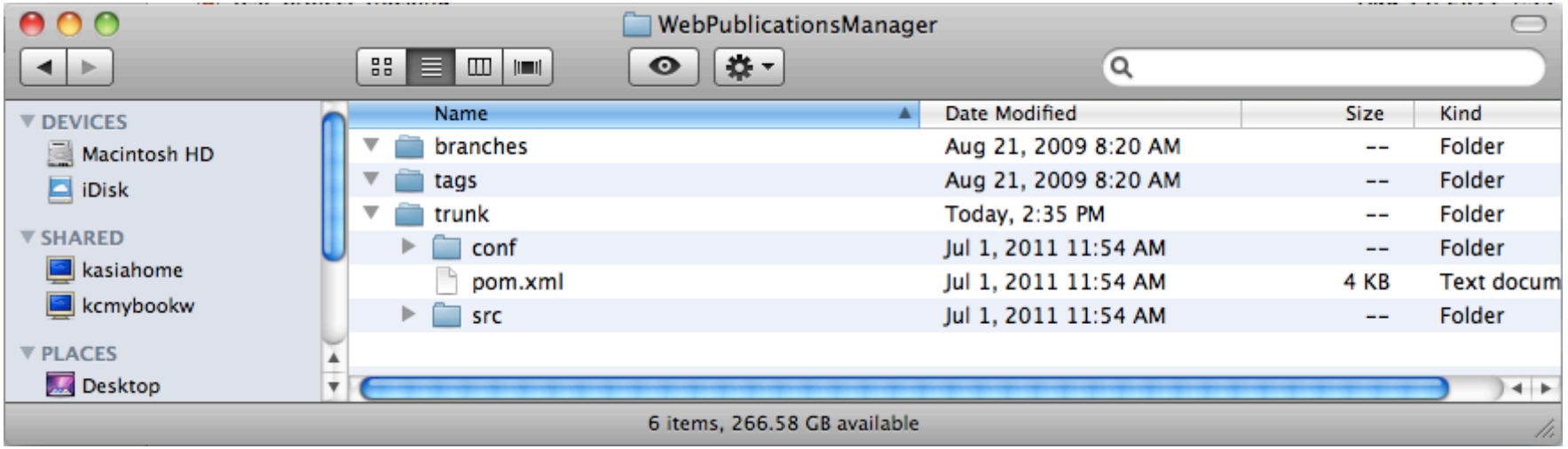
- Repositories
  - Repo – code for real
  - Sandbox – place to practice svn syntax
- Standard naming convention
  - main code in trunk
  - side/additional code in branches
  - tags are immutable

Solution to IDE-dependent builds

Initial project import

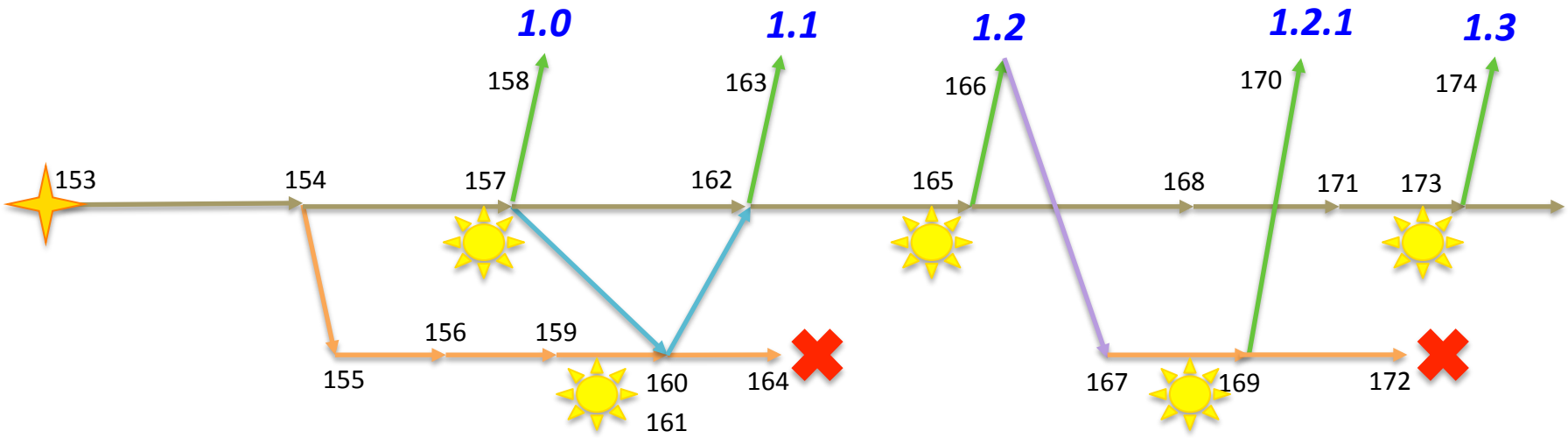


Initial project import



Solution to repository chaos

Trunk/branch/tag flow



Solution to repository chaos

- Teach, preach, and indoctrinate sound repository practices
  - Disciplined commits
  - One type of change at a time: new code, refactoring, change in directory structure, bug fix, or formatting
  - 200 lines of code or less
  - Code reviews before trunk merges
  - **NO COMMITS INTO TAGS**

Solution to repository chaos

Develop standards for commit messages

"Added some stuff"

"hmp: added methods to insert libraries for more than one platform"

"grant-management: extract method validateBudgetEntry()"

Solution to unversioned artifacts

Solution to unversioned artifacts

Tag releases before they are built and deployed

Export a tag before building

Painless builds with Apache Maven (and a great solution to missing documentation)

Apache Maven

Builds applications and libraries

Runs tests

Generates archetypes and skeletons

Generates javadocs

Apache Maven POM

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <groupId>edu.bcm.hgsc</groupId>
  <artifactId>Y-chromosome-report</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>
  <name>birt-reporting Y-chromosome report</name>
  <url>http://maven.apache.org</url>
  <dependencies>
  </dependencies>
  <build>
    <finalName>y-chromosome-report</finalName>
  </build>
</project>
```

Apache Maven POM

```
<?xml version="1.0"?>
<project  xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                              http://maven.apache.org/maven-v4_0_0.xsd" >
    <modelVersion>4.0.0</modelVersion>
    <groupId>edu.bcm.hgsc</groupId>
    <artifactId>hmp</artifactId>
    <packaging>war</packaging>
    <name>Human Microbiome Project Webapp</name>
    <version>1.4.1</version>
    <url>http://localhost:8080/hmp</url>
        <properties>
                <google.webtoolkit.home>/Library/gwt/gwt-mac-1.5.3</google.webtoolkit.home>
        </properties>
        <dependencies>
        <dependency>
            <groupId>edu.bcm.hgsc</groupId>
            <artifactId>hgsc-gwt-lib</artifactId>
            <version>hmp-maintenance-1.1</version>
        </dependency>
        <dependency>
            <groupId>com.google.gwt</groupId>
            <artifactId>gwt-servlet</artifactId>
            <scope>runtime</scope>
            <version>1.5.3</version>
        </dependency>
        <dependency>
            <groupId>com.google.gwt</groupId>
            <artifactId>gwt-user</artifactId>
            <scope>provided</scope>
            <version>1.5.3</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>myfaces</groupId>
            <artifactId>myfaces-all</artifactId>
            <version>1.1.1</version>
        </dependency>
        <dependency>
            <groupId>struts</groupId>
            <artifactId>struts</artifactId>
            <version>1.2.4</version>
        </dependency>
```

Apache Archiva
Solution to missing artifacts

Hudson
Continuous integration server

Solution to builds that break

Solution to builds that break

- Maintain consistent dependency versions in pom.xml
  - as long as dependencies don't change, build won't break
- Continuous integration server performs builds in a standardized environment
  - builds are not dependent on individual programmers' machines
- Make a branch for integration and check it out
- Update dependencies in in your working copy when you are ready integrate
  - first unit tests will be run on a local machine

Solution to builds that break

As changes are made in application code, update dependency version in pom.xml and commit your code to trigger tests run by continuous integration server

you should be seeing fewer and fewer tests failing with each new commit

When all features are ready and all tests passed: tag, build, deploy, and enjoy

Solution to unreadable code

Solution to unreadable code

Develop and enforce code standards

- Indents
- Number of characters per line
- Naming convention for classes and methods
- Number of empty lines between methods
- Comments

One-step build and tag process

One-step build and tag process

```
mvn test-compile clean
$EDITOR pom.xml
rm pom.xml~
echo 'ENTER PROJECT NAME:'
read project
echo 'ENTER VERSION #:'
read version
echo 'ENTER THE NEW SNAPSHOT #:'
read snapshot
svn ci -m"BCM $project release $version”
mvn package source:jar
ls target
echo 'COPY THE BINARY JAR NAME FROM ABOVE AND PASTE TO THE COMMAND LINE:'
read binaryarchive
internal-deploy target/$binaryarchive pom.xml
```

One-step build and tag process

```
echo 'COPY THE SOURCE JAR NAME FROM ABOVE AND PASTE TO THE COMMAND LINE'
read sourcearchive
internal-deploy target/$sourcearchive pom.xml -Dclassifier=sources
svn cp $SVN_LOCATION/trunk $SVN_LOCATION/tags/project-$version
mvn javadoc:javadoc
mkdir  /Library/WebServer/Documents/javadocs/$project/
mv target/site/apidocs/ /Library/WebServer/Documents/javadocs/$project/$version/
$EDITOR pom.xml
rm pom.xml~
svn ci –m “BCM $project release $version pom => $snapshot –SNAPSHOT”
mvn clean
```

Resources

http://subversion.apache.org/

http://maven.apache.org/

http://archiva.apache.org/

http://hudson-ci.org/