CERN School of Computing

Iterative Development

Brice Copy
Sebastian Lopienski

CERN

What Is Iterative Development ?

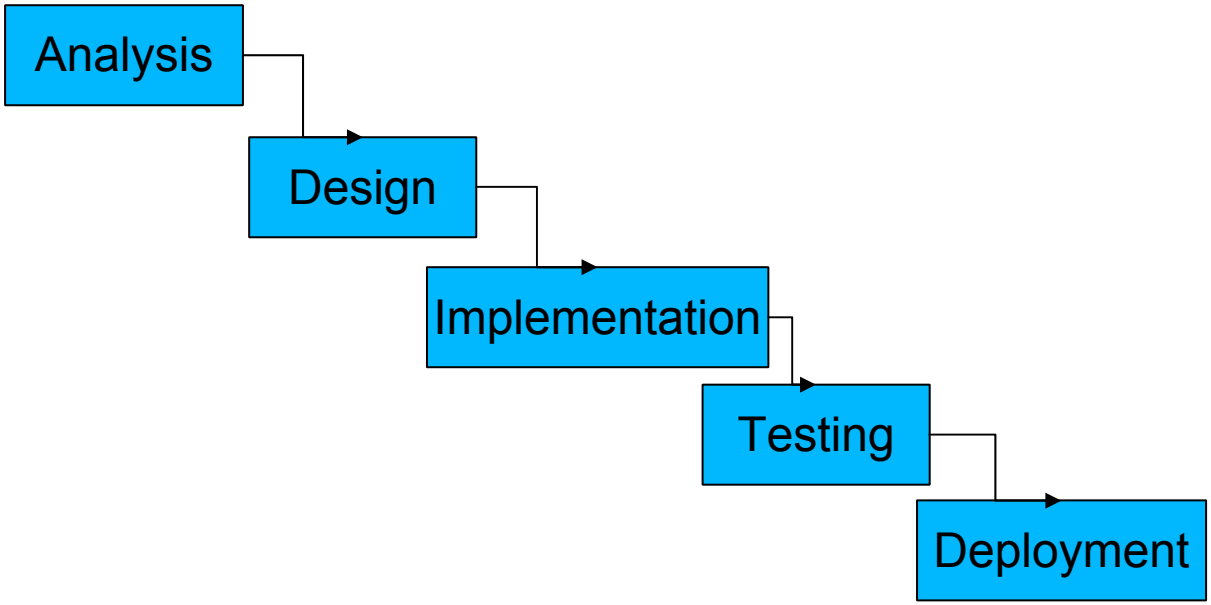
- Perform full, fast and complete development cycles (spec, code, build, integrate, test and back again)
- In line with modern risk management techniques
- Enables you to cope with changing requirements
- As opposed to monolithic approaches (cascade model)

Lecture overview

- Defining iterative development, its uses, its benefits
- How to implement it for your projects, with focus on :
  - Configuration Management (or Change Management) Tools - *(S. Lopienski)*
  - Integrated Builds – *(B. Copy)*

Cascade Model

- Already identified the need for a process (spec, code, build, integrate, test and back again)
- Suitable for small projects



CERN School of Computing

Why Iterative Development Was Introduced

- Cascade development too cumbersome
- It addresses greater risks first
- It is “fail fast” - too many IT projects fail at the very end (when all the money is spent)
- Full development cycles let your team members (Dev, QA, System) work in parallel

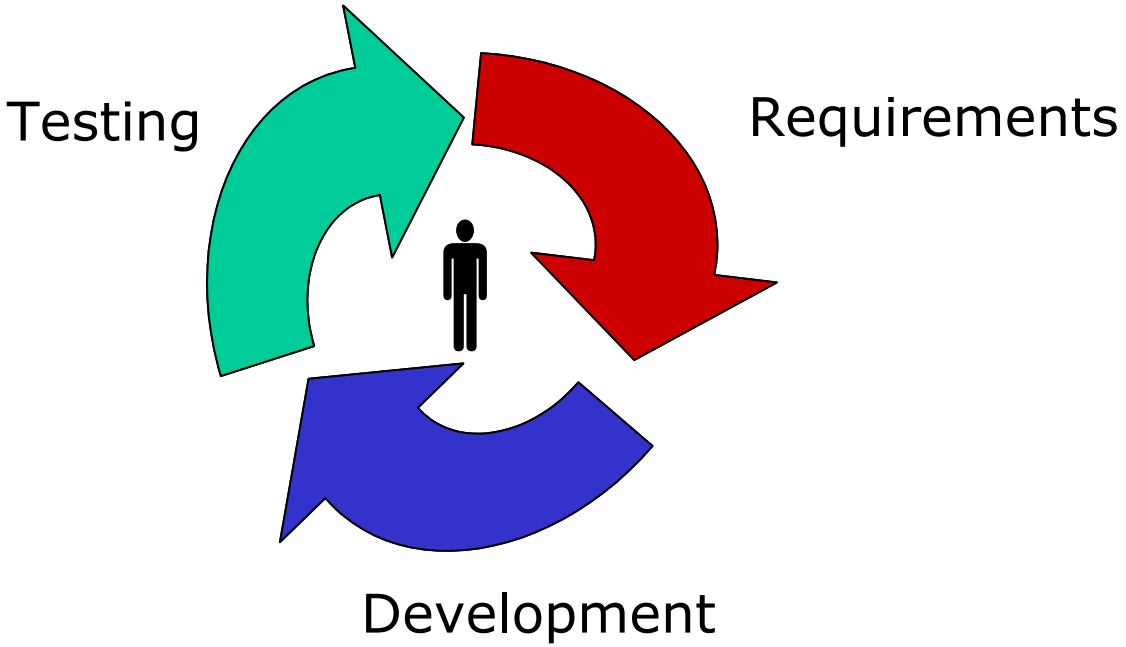
Where Is It Used

- Microsoft
  - Windows NT was the first large software product built and integrated on a daily basis
  - Yielded a stable product (NT 4) and largest hardware support (16.5 millions LoC)
- Oracle
  - Agile style of development is used for making developer tools (such as JDeveloper)
  - Daily builds with full QA cycles
  - Other metrics to monitor health of the project (outstanding bug count, failed tests...)

Where Is It Used (continued)

- Open source projects
  - More and more large projects rely on continuous builds (Spring framework, Apache, JBoss)
  - Teams are geographically spread, SCM server is their main collaboration tool
- CERN
  - In order to cope with change
  - Resources are limited for “background” tasks
    - QA
    - Documentation
    - Release scheduling and planning

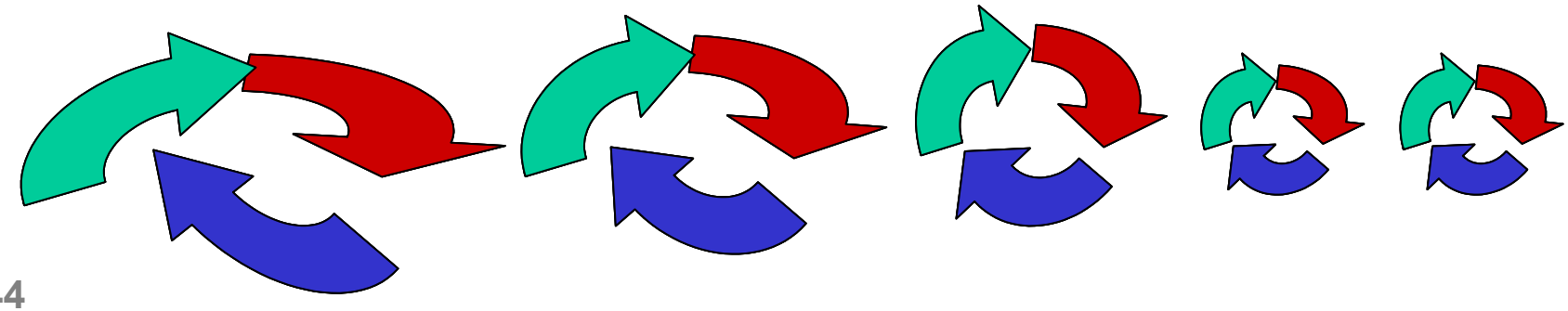
The three phases



CERN School of Computing

Progression

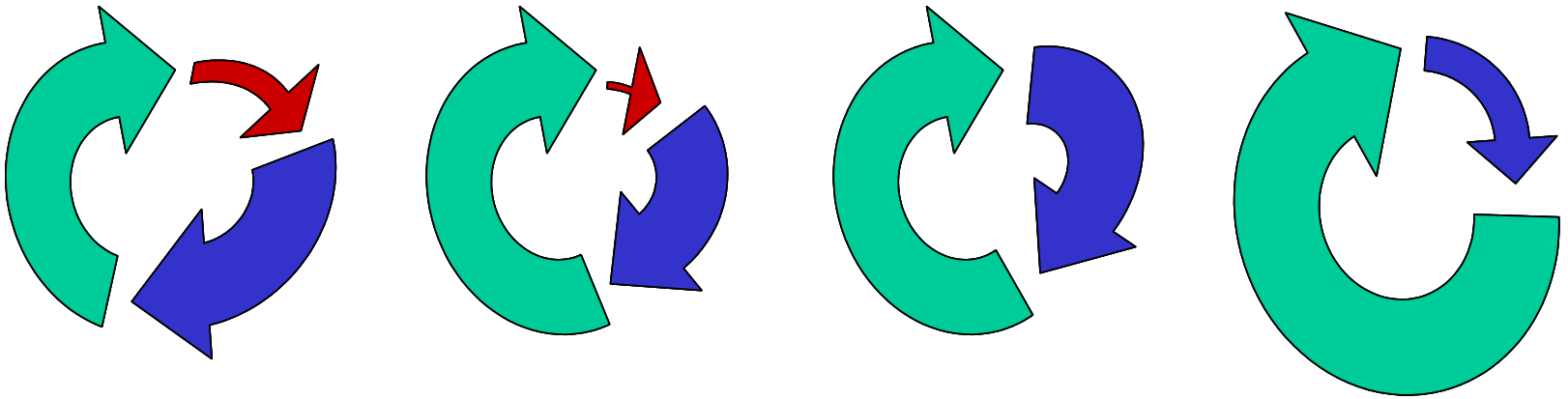
- Initial cycle are longer (a couple of weeks)
- No prototype is usually delivered before the second iteration
- Cycles get shorter and shorter as the project progresses
- When necessary features are provided – focus on quality



CERN School of Computing

Progression (2)

- Product Management gets more and more quiet
- Development pressure increases
- Quality takes more and more importance
- Eventually, Quality dictates Development, which must deliver punctual improvements and in the end just bug fixes



“Et pour la pratique”

Gotta love the theory...
but who will apply it and how ?

Focus on :

- Change Control
- Iterative Builds

Best practices policy

- To work as a team, you need to define your best practices (in order of importance) :
  - SCM practices (branching, tagging, commits)
  - Testing practices
  - Dependency management (ensure convergence)
  - Coding standards and review processes etc...
- Communicate and agree on those, best practices are not a one man's job
- Tip : If you do not have policies, steal them from someone (they won't mind)

Configuration Management

a.k.a. Change Management

a.k.a. “The fall guy”

- Monitoring change in *i*terative development is paramount
- Being able to produce a deliverable from “the good old days when everything worked fine”
- Focus on CVS : Popular Software Configuration Management (SCM) tool

Advanced CVS features

- Starting point : CSC 2004 - CVS usage lecture
- Here are some advanced features helpful for teamwork :
  - Tagging
  - Branching
  - Merging
  - Watching

CERN School of Computing

Tagging

- Giving a common name to chosen revisions of chosen files
- Useful to mark a release made at a given moment (“current revisions of all files”), to mark a project as it is at the given time
- You can later refer to that tag (name) while checking out, branching and merging etc.

```
cvs tag Tag_Name
```

tags current revisions of files

Branching

- Branch : separate thread of revisions, that can be edited without affecting other branches
- Useful for maintaining latest stable release without touching current development (unstable) version
- If several developers have to modify one file, each should work on his branch

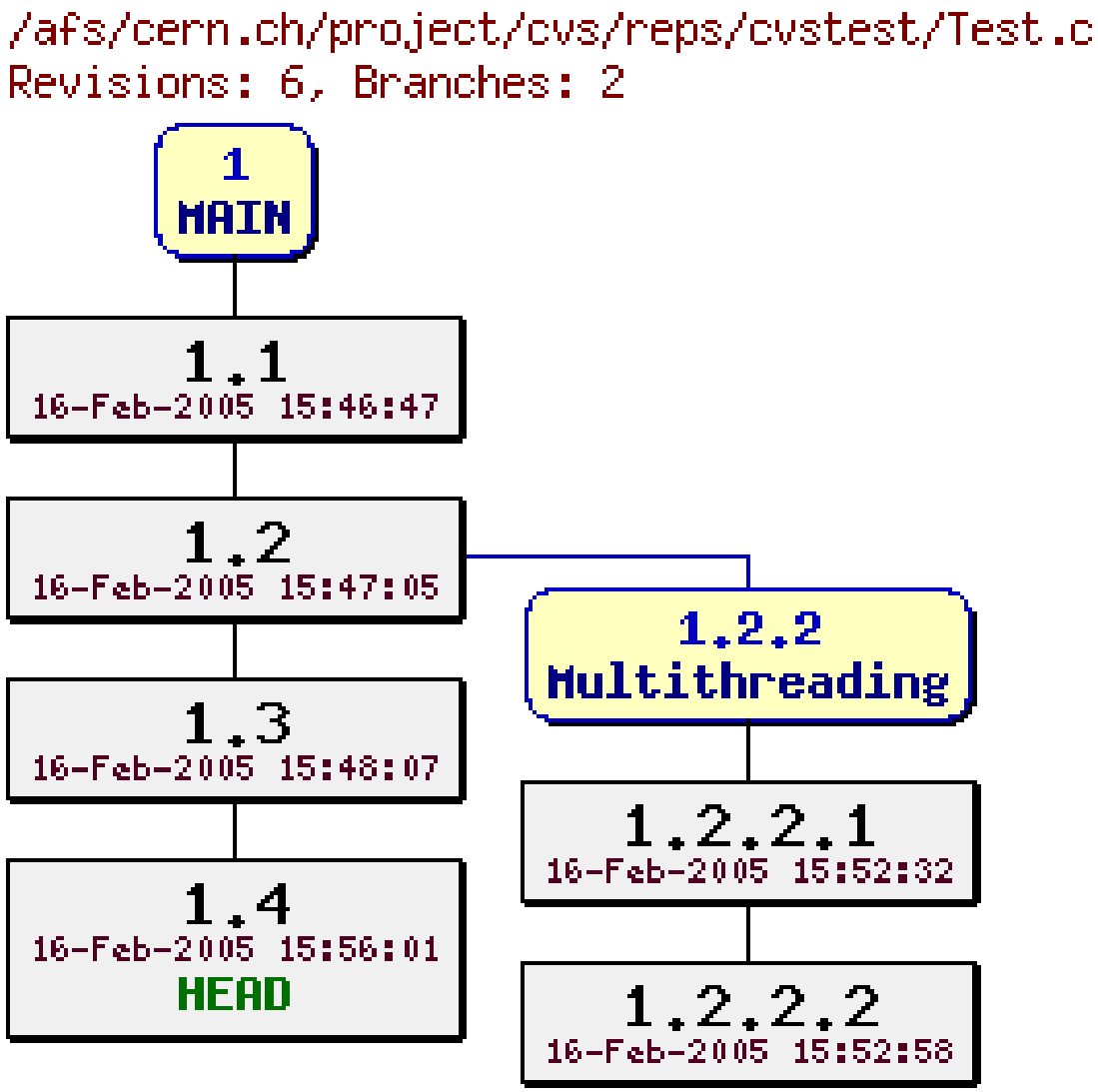
  `cvs tag -b Branch_Name`
  (creates a new branch)

  `cvs update –r Branch_Name`
  (updates local working copy)

- Sample branch number 1.5.2.1

  = *first revision 2.1 of a branch made from revision 1.5*

Branching : revision tree



Branching cost

- Branching is a powerful feature
- Like all powerful features it comes at a cost :
  - Branching means maintaining multiple versions of your product
  - You may have to fix bugs only in a given branch
  - You may have to fix bugs in all branches (can be difficult or impossible in some cases)
  - A branch should be as short lived as possible

CERN School of Computing

Merging

- It is closing a branch by putting its modifications into the mainstream “trunk”
- Or merging modified local copy of a file with modified revision in CVS
- CVS tries to merge modifications automatically
- if it fails because of a conflict (same line was modified in a branch and in a “trunk”), then developer has to merge it manually

```
cvs update -j Branch_Name
```

“joins” changes of the other branch

CERN School of Computing

Watching

- When a developer sets a watch on a file, he asks CVS to notify him if anyone else starts to work on that file

`cvs watch add File_Name`
asking CVS to watch this file for me

`cvs edit File_Name`
informing CVS that I start working on this file

`cvs unedit File_Name`
I'm not working on this file anymore

`cvs watchers File_Name`
who is watching this file?

CVS Tools

- Beyond the command line
  - GUI CVS clients
  - Web CVS client
- Let you :
  - Visualise and edit differences between versions
  - Request revision trees
  - Perform advanced operations easily (Special updates by date, tag, branch)

CVS Tools samples

Current directory: [acc-co] / accsoft / commons / accsoft-commons-cache

Files shown: 3

File	Rev.	Age	Author	Last log entry
src/				
build.xml	1.2	5 days	lmestre	converted to release
people	1.1	5 days	lmestre	converted to release
product.xml	1.3	5 months	vbaggiol	corrected dependency on a

Show files using tag: - Non-branch tags - Show

Diff for /accsoft/gui/accsoft-gui-frame/product.xml between version 1.2 and 1.3

version 1.2, 2004/09/21 13:02:32	version 1.3, 2004/09/29 16:34:25
Line 1	**Line 1**
<?xml version="1.0" encoding="UTF-8"?>	<?xml version="1.0" encoding="UTF-8"?>
<products>	<products>
<product name="accsoft-gui-frame" version="0.0.1" directory="accsoft/gui/accsoft-gui-frame">	<product name="accsoft-gui-frame" version="0.0.2" directory="accsoft/gui/accsoft-gui-frame">
<desc>GUI frame based on JDNC</desc>	<desc>GUI frame based on JDNC</desc>
<href></href>	<href></href>
<dependencies>	<dependencies>
<dep product="jdnc"/>	<dep product="jdnc"/>
	<dep product="accsoft-commons-core"/>
<dep product="commons-logging"/>	<dep product="commons-logging"/>
<dep product="log4j" local="true"/>	<dep product="log4j" local="true"/>
</dependencies>	</dependencies>

Legend:
Removed from v. 1.2
changed lines
Added in v. 1.3

Colored Diff Show

CERN School of Computing

Once upon a time...

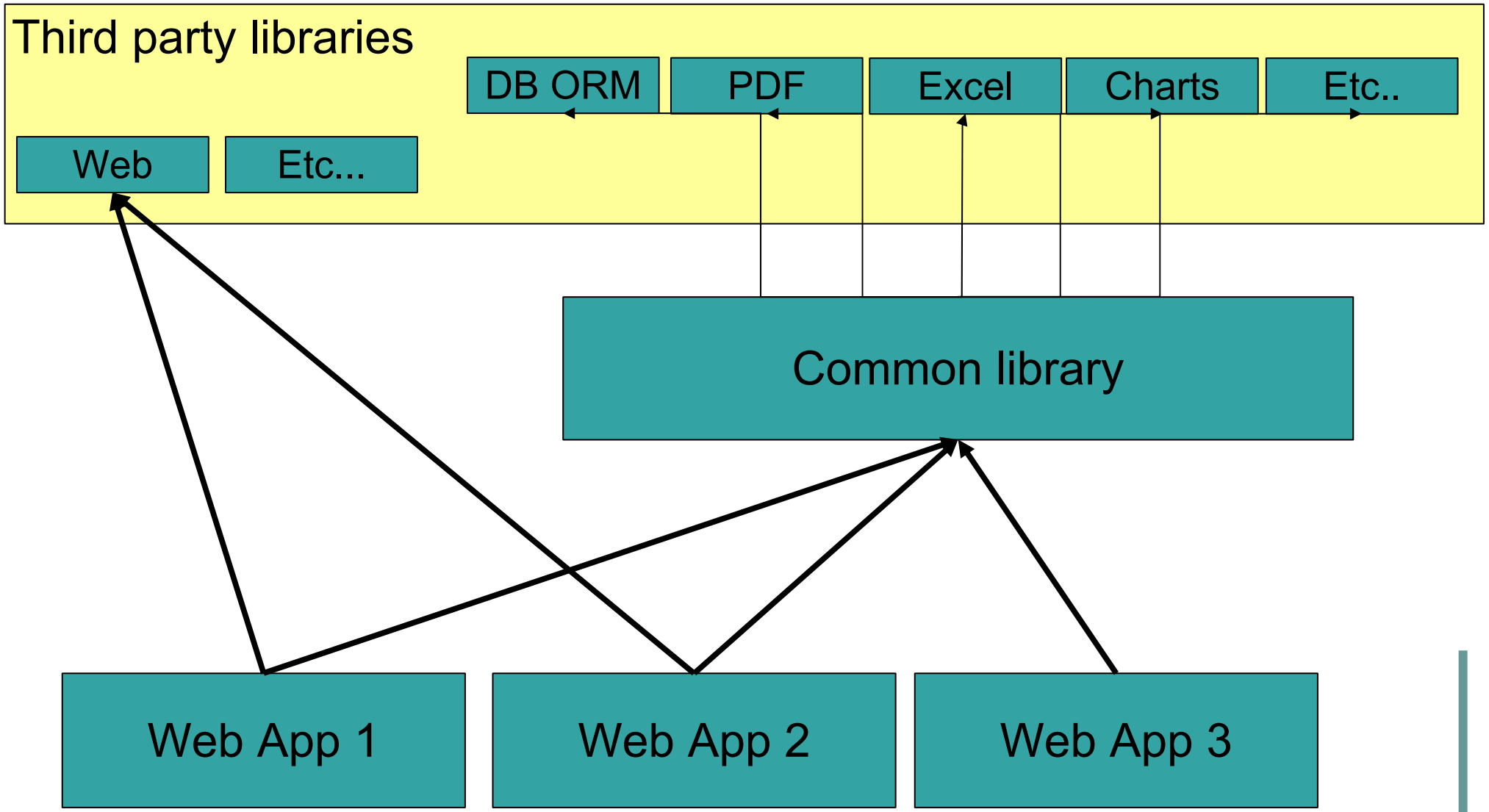
or “The three developers and the big bad build”

- A team of developers sitting on a java web application :
  - A big common library (for foundation classes)
  - A big application made of :
    - A set of disconnected CVS modules and deployed separately (for reusability)
    - Web UI made of JSP pages
    - Many third party dependencies = Feature rich
  - Manual testing procedure
  - Manual configuration and deployment

CERN School of Computing

Once upon a time...

Dependencies



CERN School of Computing

Once upon a time...

Build troubles

- Building from scratch was difficult
  - Dependencies version number was not known (difficult upgrades), lived in one place only
  - Near the end : the common library needed to be compiled by bootstrapping (A→B→A)
- Configuring for deployment required a global understanding of the product (config files in multiple places)
- Deploying needed a manual procedure
- The end result was tested visually

CERN School of Computing

Once upon a time...

The integrated build

- Integrated build helped to :
  - Break up the common library in small components with few dependencies
  - Ensure the end-product could be built from scratch by anybody
  - Make it easy to write tests and run them continuously
  - Collect metrics on development activity
- Integrated build did not :
  - Write tests automatically
  - Fully automate the deployment

CERN School of Computing

Why so extensive ?
“Your build”

- Your build must be :
  - Reproducible
  - Easy to trigger (one command line)
  - Automatable
- Your build must cover all aspects of your development procedure
- Your build must run as early and as often as possible (you only care when it's broken)

CERN School of Computing

Integrated Build Tool (1)

What does it do ?

- Code Generation
  - Metadata, Remote stubs, ORM mapping files
- SCM integration
  - CVS, Subversion, SourceSafe etc...
- Code compilation (from various sources to various targets)
  - Functional and regression testing
  - Packaging (ZIP/RPM, JAR/WAR/EAR files)
- ...

CERN School of Computing

Integrated Build Tool (2)

What does it do ?

- Testing
  - Functional, Regression, Integration...
- Packaging and deployment
  - ZIP, RPM, JAR/WAR/EAR etc...
- Documentation generation
  - Javadoc, XDOC, UML, etc...
- Reporting
  - CVS activity statistics, unit testing coverage, code quality metrics
- And more...

Which build tools ?

- Apache Ant
  - All purpose tool, low level
- Apache Maven
  - High level, somewhat Java centric
- Cruise Control
  - For build automation
- But there are many more out there...

CERN School of Computing

Apache Ant

- Aimed at replacing MAKE
- Low level tasks (move, zip, javac etc..)
- Project organisation is up to you
- Making new tasks is easy...
- ...Sharing them is not easy
- Will not manage your project (needs strong processes or a generation tool)
- Good foundation for platform independent build processes and scripting

Ant build sample

```
<project name="jpetstore" default="dist" basedir=".">
    <target name="init">
      <path id="project.classpath">
          <fileset dir="${global.build.dir}/comp">
              <include name="log4j/lib/log4j.jar"/>
              <include name="junit/lib/junit.jar"/>
          </fileset>
      </path>
      <available file="${dir.src}/java"
  property="sources.exist"/>
  </target>

  <target name="compile" depends="init" if="sources.exist">
    <mkdir dir="${dir.build}/classes"/>
    <javac debug="${debug}" destdir="${dir.build}/classes"
   srcdir="${dir.src}/model">
       <classpath refid="project.classpath"/>
    </javac>
  </target>
</project>
```

CERN School of Computing

Apache Maven

- A layer on top of Ant
- Includes a project model (=metadata)
- Requires a reorganisation of your dependencies
- Uses Ant tasks, scripting and plug ins
- Covers all steps of your build (from code generation to deployment)
- Really aimed at Java (but offers .Net plug ins for compilation and code generation etc...)

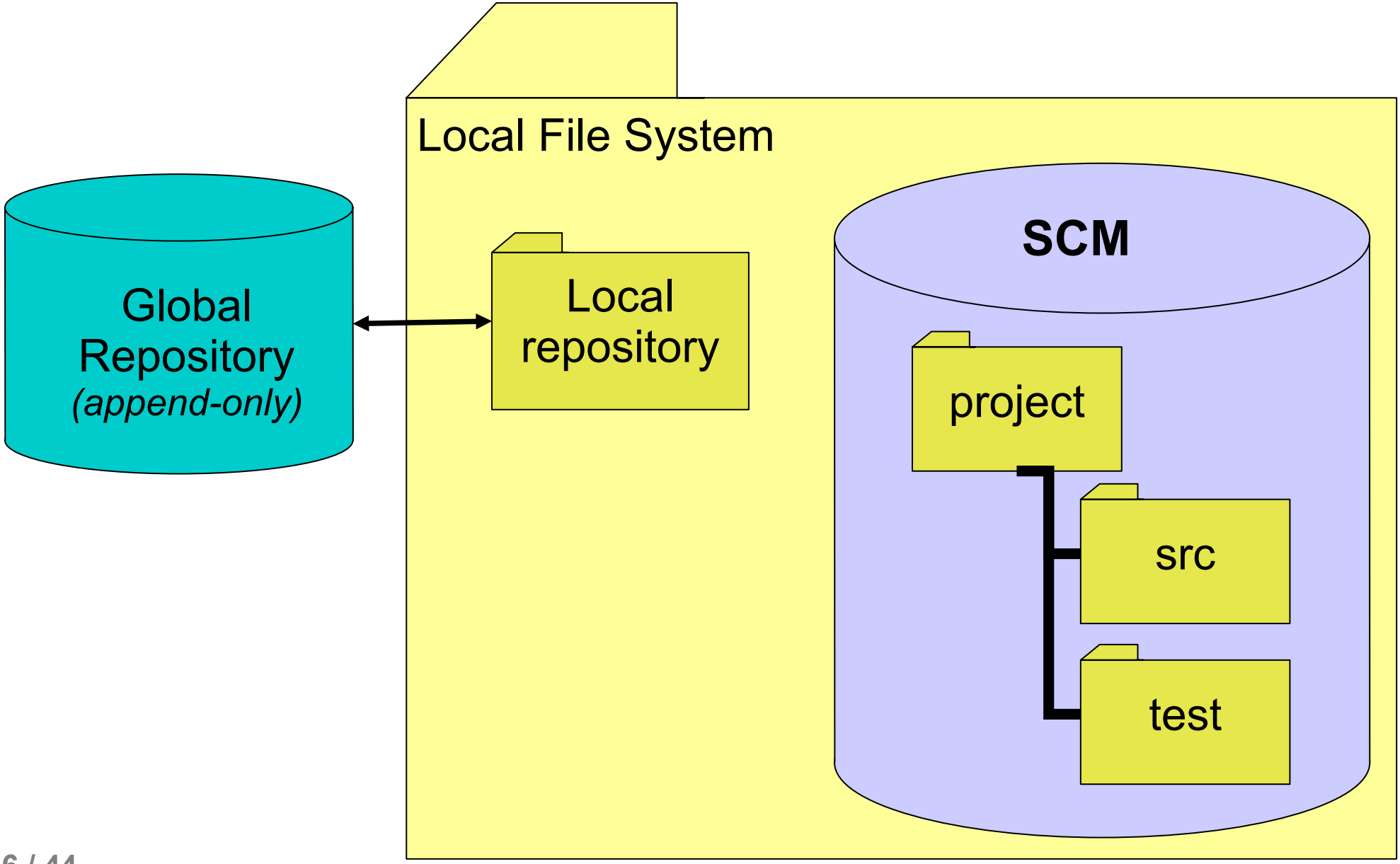
Maven Project Model (POM)

- Requires you to describe :
  - Your source files and resources
  - Your dependencies (JAR, WAR, ZIP etc...)
  - Your SCM connection (CVS, Starteam, Subversion...)
- Gives the exact recipe for a reproducible build
- Lets you define custom build steps that decorate existing steps
  (*e.g. "Before compilation -> trigger this generation utility"*)

Maven features

- In return, your project can now be :
  - Generated
  - Compiled
  - Tested
  - Packaged
  - Deployed
- ... all this with a single command line
- Maven will also generate reports (CVS stats, code quality, javadoc, xdoc, testing coverage)

Maven project layout



Maven project file sample

```
<project>
  <name>Pet Clinic</name>
  <groupId>cern.ppt</groupId>
  <id>petclinic</id>
  <currentVersion>0.1</currentVersion>

  <package>org.springframework.samples.petclinic</package>

  <dependencies>
    <dependency>
      <groupId>hibernate</groupId>
      <artifactId>hibernate</artifactId>
      <version>2.1.7</version>
      <properties>
        <war.bundle>true</war.bundle>
      </properties>
    </dependency>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <unitTestSourceDirectory>test</unitTestSourceDirectory>
  </build>
</project>
```

Maven output samples

Dashboard report

Column legends

Project	JCoverage %lines	JCoverage LOC	JUnit Errors	JUnit Failures	JUnit Pass Rate
Model	4%	5900	26	51	39%
UI	22%	2108	0	11	60%

Coverage report

	Files	%line	%branch
Project	18	48%	56%

Packages

cern.ppt.download	4	67%	69%
cern.ppt.download.dp	3	74%	90%
cern.ppt.download.dp.bc4j	3	0%	0%
cern.ppt.download.dp.beans	1	0%	0%
cern.ppt.download.dp.xml	1	95%	97%
cern.ppt.download.render	3		
cern.ppt.download.render.excel	1		
cern.ppt.download.util	2		

cern/ppt/download/dp/beans/BeanDataProvider.java

Violation	Line
Avoid unused imports such as 'Vector'	46
Avoid unused private fields such as 'm_bKey'	100
Avoid unused local variables such as 'dynaProp'	213

Continuous builds

- Continuous builds are like watchdogs
- Take the pain out of building code
- Send daily status messages
- Keep log archives, to help you monitor your progress
- Inform whoever last contributed that there's a problem

CERN School of Computing

Cruise Control

- Continuous build tool
- Very simple to install and run
- Works with many building tools (Ant, Maven, NAnt)
- Publishes results via :
  - Email
  - Scp
  - Instant Messaging
  - X10 (Heating control, lava lamp, alarm etc...)

Cruise Control report sample

CERN School of Computing

BUILD FAILED

Ant Error Message: E:\Projects\cvs\cruisecontrol\main\sample_project\build.xml:75: Compile failed, messages should have been provided.

Date of build: 20020507023938

Time to build: 6 seconds

Last changed: 05/07/2002 04:25:33

Last log entry:

Errors/Warnings: (7)

```
E:\Projects\cvs\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java:7: illegal start of expression
        ?
        ^
E:\Projects\cvs\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java:7: ';' expected
        ?
         ^
2 errors
```

Unit Tests: (1)

All Tests Passed

Modifications since last build: (1)

change User E:\Projects\cvs\cruisecontrol\main\sample_project\src\java\hello\HelloWorld.java/HelloWorld.java

Iterative = Integrated

- For iterative development you need
  - The right tools
  - The right practices
  - The right project model
- Do not focus on a tool, but on what you really need
- Iterative Development is contagious – once you start somewhere, the rest of your projects have to follow

CERN School of Computing

And to follow up...

- Q&A
- Semi-interactive demo on build integration
- Panel discussion

CERN School of Computing

Bibliography
Recommended links

- Pragmatic Project Automation
  by M. Clark *(Pragmatic Bookshelf, July 2004)*
- The resource on agile / iterative development
  http://www.agilealliance.org/articles/index
- Testing practices blog
  http://www.developertesting.com/
- *Maven User Reference*
  http://maven.apache.org/reference/user-guide.html