Large-Scale Software Development

Lecture 2 :Communication, Contribution, Architectures and Tools

LINKÖPINGS UNIVERSITET

Agenda

- Communication, contributions
- Seminars
- Coding conventions, design principles and high-level structures
- Tools

Communication

- the imparting or exchanging of information by speaking, writing, or using some other medium.

- means of sending or receiving information, such as telephone lines or computers.

Contribution

- a payment (as a levy or tax) imposed by military, civil, or ecclesiastical authorities usually for a special or extraordinary purpose

- the act of contributing

Participate

- to take part
- to have a part or share in something

Participate

Contribution

Communicate

Seminar 1: A Project (well actually 2)

- The Community
    - Owner
    - Maintainers and Collaborators
    - Contributors
    - Community Members
- Documentation
    - README
    - CONTRIBUTING
    - LICENSE
    - Wiki
    - IRC

How to contribute (before coding)

- Report a bug (Issue)
- Verify and track down a reported bug
- Write an example for the documentation
- Edit the project wiki
- Answer a question
- Blog about your experiences (for open source)

Contributing: Issues

- Check existing issues
- Be clear
- Link to demos (demonstrate the issue)
- Include system details
- Paste error output

Fixing an issue (pull request)

- Fork the repository and clone it locally.
    - Pull in changes from ‘upstream’ often so that you stay up to date.
- Create a branch for your edits.
- If a bug fix: Be clear about what problem is , explain how to reproduce it
- Test test test: Run your changes against any existing tests if they exist and create new ones when needed.
- Contribute in the style of the project to the best of your abilities
- Open a Pull Request : Wait for review

Future seminars

- Seminar 2
  - Investigate the software design
- Seminar 3
  - Become developer for/in your project

Part 2

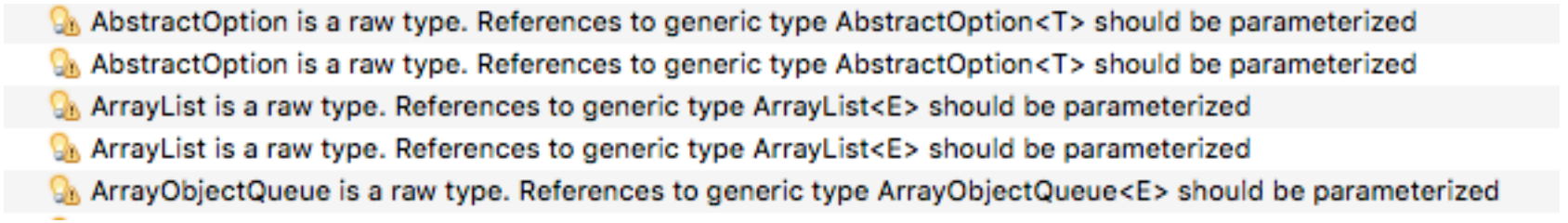
- Code conventions - design principles - high-level structures

Code conventions

```
def loadSchemas():
  schemaTuples, badSchemaFiles = loadAllJsonObjects("schemas")
  schemas = {}
  for path, fileName, o in schemaTuples:
    schemas[fileName[:-5]] = o
  return schemas, badSchemaFiles
```

Syntax and style

```
def load_schemas():
    schema_tuples, bad_schema_files = load_all_json_objects("schemas")
    schemas = {}
    for path, file_name, o in schema_tuples:
        schemas[file_name[:-5]] = o
    return schemas, bad_schema_files
```



Compiler warnings

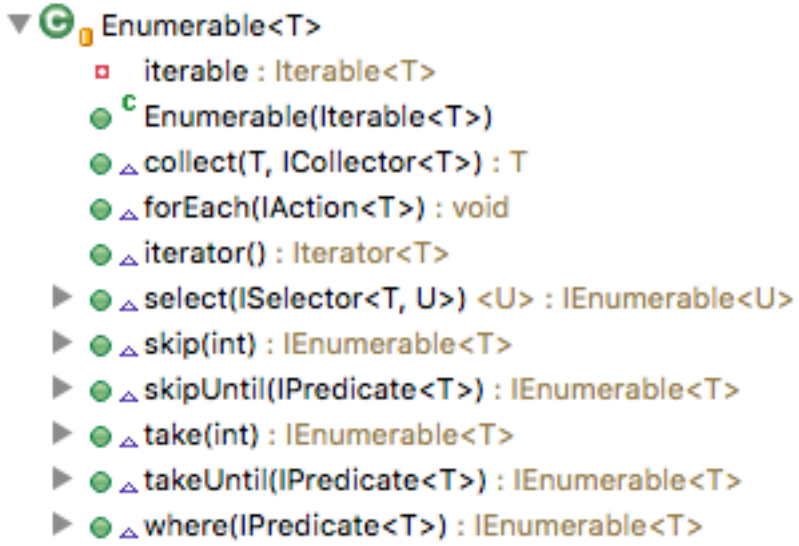
```
int[] a = new int[] {1,2,3};
int sum=0;
for (int i : a) {
    sum+=a[i];
}

Arrays.stream(a).sum();
```

Language constructs

Design principles

```
/** Constructor of the Cell class.
 *  It is the only way to set the name of the cell.
 *  @param name the name of the cell
 */
public Cell(String name) {
    this.name = name;
    formula = "";
    observers=new LinkedList<CellObserver>();
    anal = Analyzer.getInstance();
    state = EquationState.getInstance();
    astN = new ASTValue(0.0);
```

Coupling

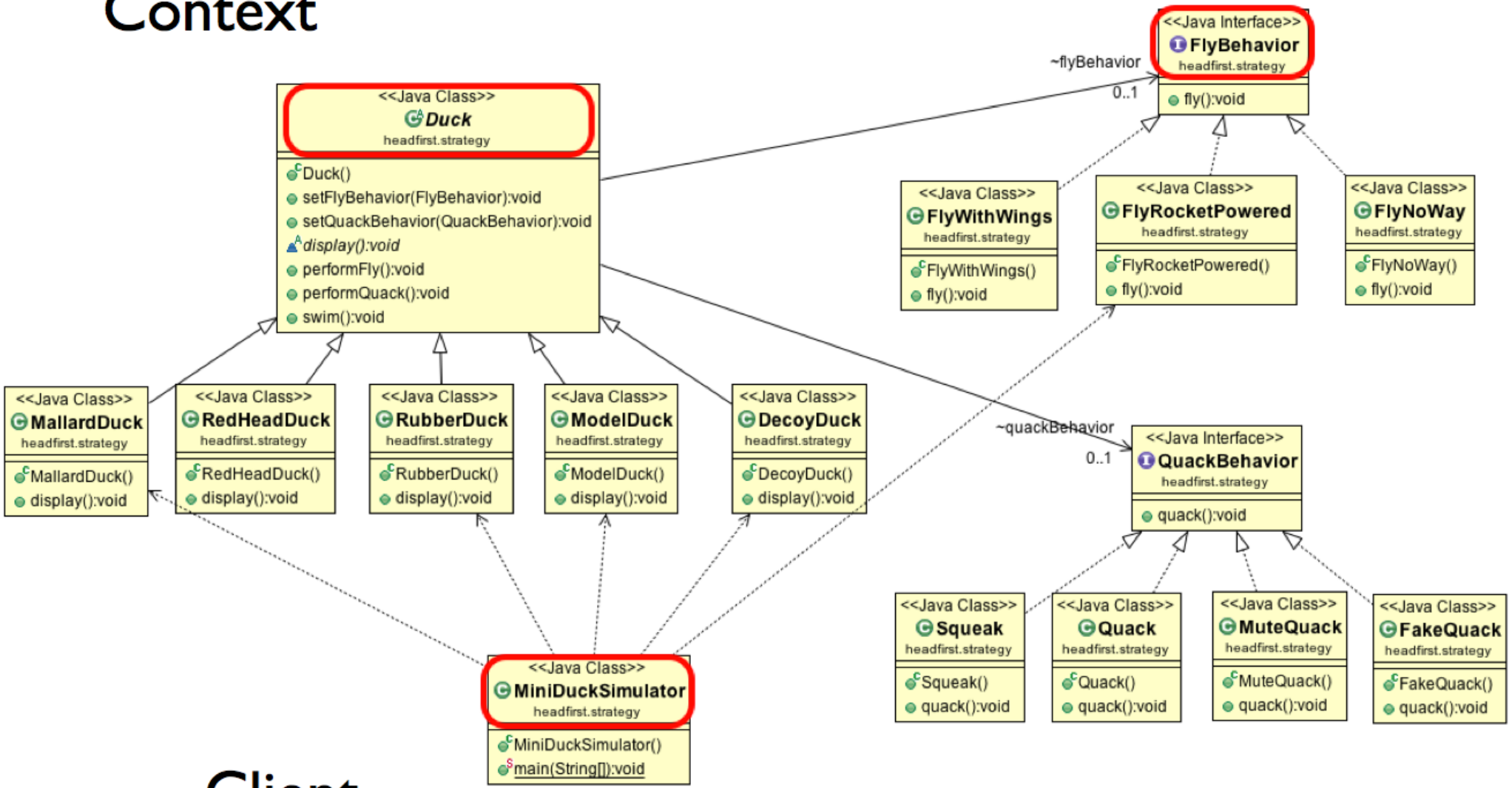


Cohesion

Strategy

Context

Strategy



Client

Software architecture

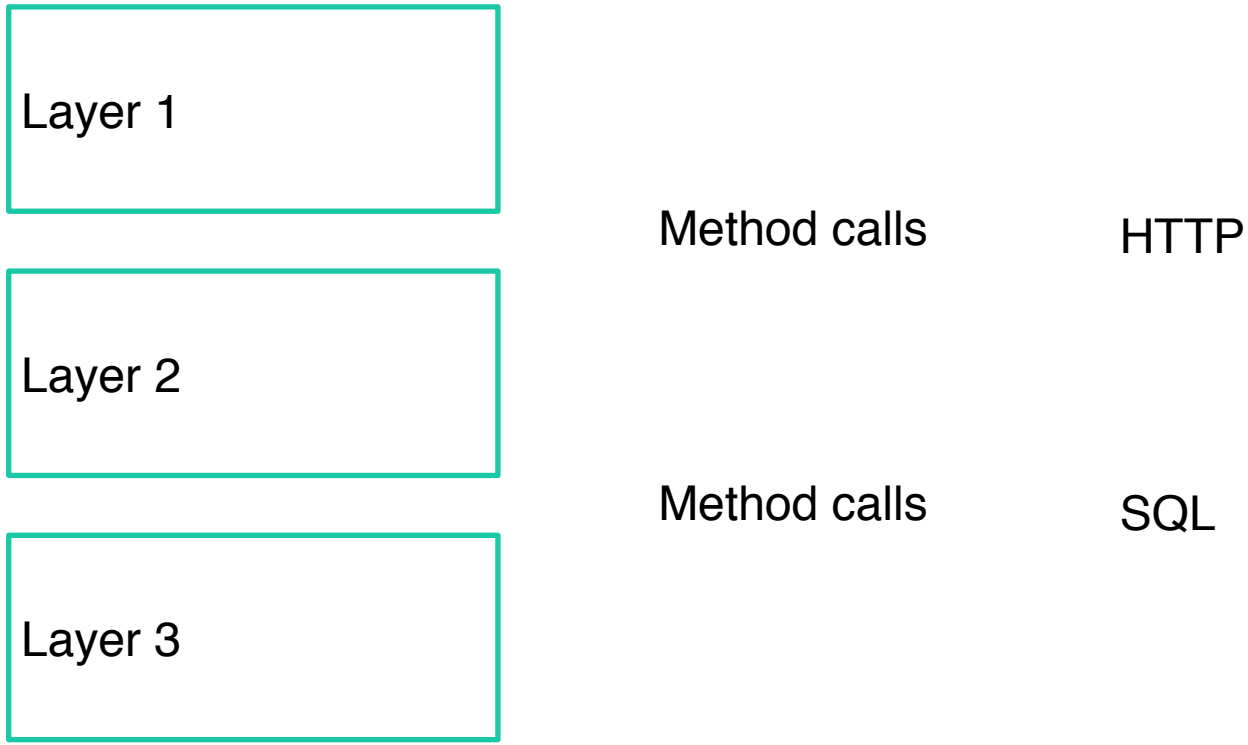
- ”*The set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both*” - Documenting Software Architectures: Views and Beyond (2nd Edition), Clements et al, Addison-Wesley, 2010
- ”*the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution*” - ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems
- ”*That peculiar consistency and homogeneity of Design that would give rise to ease of use and reuse, which is wished for by the Developer who calls himself "The Software Architect.*”” - John Carter (Software Technologist, Tait Electronics, Christchurch, New Zealand)

https://www.sei.cmu.edu/architecture/start/glossary/definition-form.cfm

Software architectures

- Flexibility - estimated cost of making changes to individual parts
- Testability - estimated cost of devising tests for parts or the whole
- Scalability/Performance - estimated cost of changing the system to accommodate higher loads
- Cost of development - estimated cost of developing the common core
- Cost of deployment - estimated cost of running the system

Layered architecture



Spring PetClinic

GitHub, Inc. [US] | https://github.com/spring-projects/spring-petclinic

Spring PetClinic Sample Application build passing

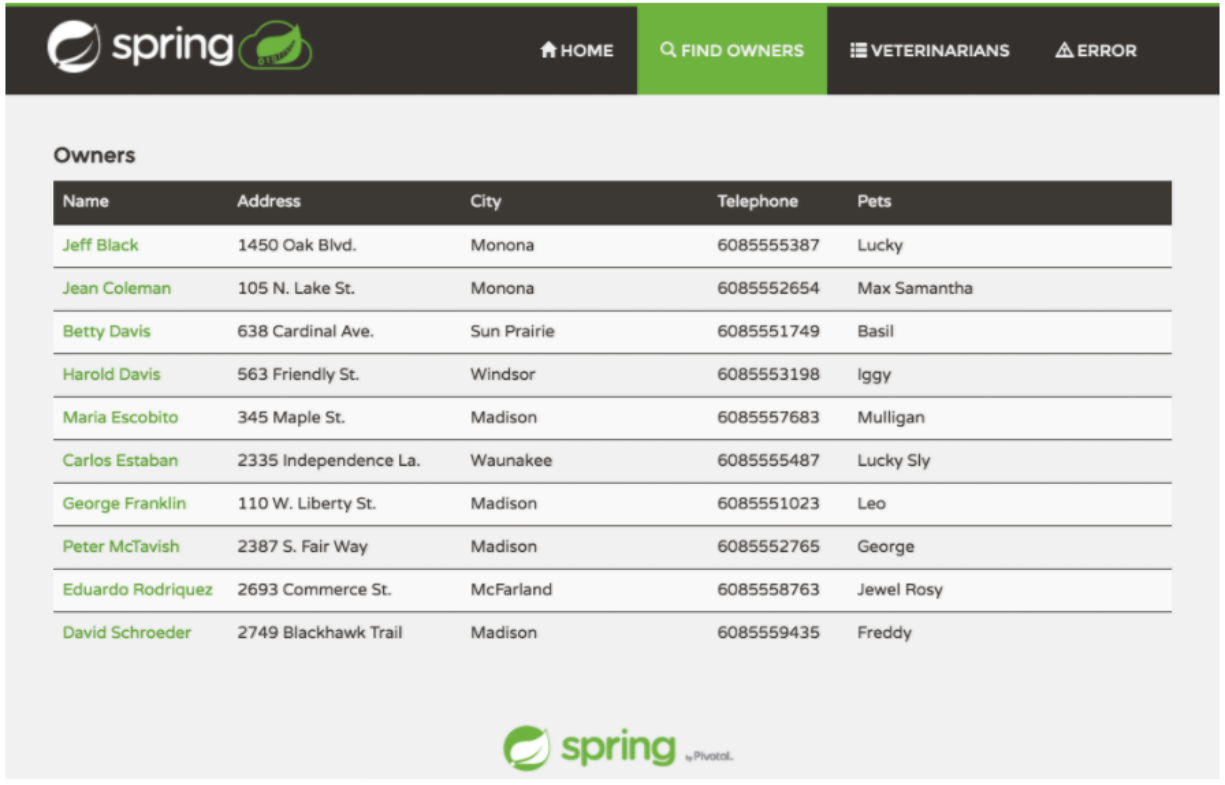
Understanding the Spring Petclinic application with a few diagrams

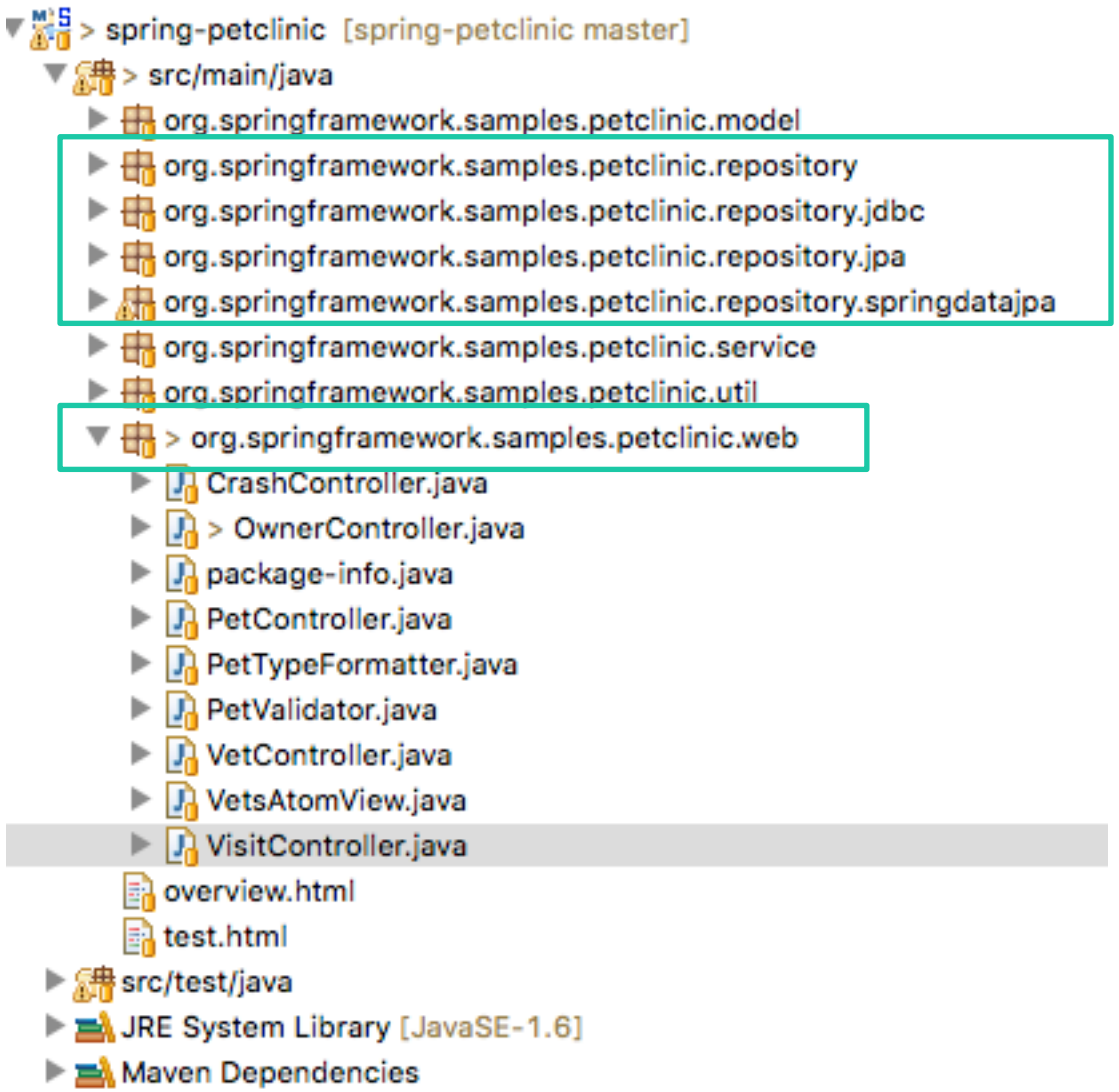
See the presentation here

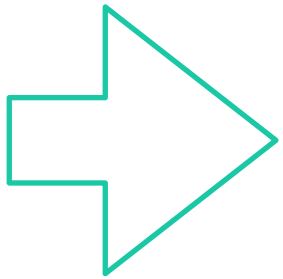
Running petclinic locally

```
git clone https://github.com/spring-projects/spring-petclinic.git
cd spring-petclinic
./mvnw spring-boot:run
```

You can then access petclinic here: http://localhost:8080/




spring-petclinic [spring-petclinic master]
src/main/java
org.springframework.samples.petclinic.model
org.springframework.samples.petclinic.repository
org.springframework.samples.petclinic.repository.jdbc
org.springframework.samples.petclinic.repository.jpa
org.springframework.samples.petclinic.repository.springdatajpa
org.springframework.samples.petclinic.service
org.springframework.samples.petclinic.util
org.springframework.samples.petclinic.web
CrashController.java
OwnerController.java
package-info.java
PetController.java
PetTypeFormatter.java
PetValidator.java
VetController.java
VetsAtomView.java
VisitController.java
overview.html
test.html
src/test/java
JRE System Library [JavaSE-1.6]
Maven Dependencies



```java
public class VisitController {

    private final ClinicService clinicService;


    public VisitController(ClinicService clinicService) {

    public void setAllowedFields(WebDataBinder dataBinder) {

     *  Called before each and every @RequestMapping annotated method.
    @ModelAttribute("visit")
    public Visit loadPetWithVisit(@PathVariable("petId") int petId) {
        Pet pet = this.clinicService.findPetById(petId);
        Visit visit = new Visit();
        pet.addVisit(visit);
        return visit;
    }

    // Spring MVC calls method loadPetWithVisit(...) before initNewVisitForm is called
    @RequestMapping(value = "/owners/*/pets/{petId}/visits/new", method = RequestMethod.GET)
    public String initNewVisitForm(@PathVariable("petId") int petId, Map<String, Object> model) {
        return "pets/createOrUpdateVisitForm";
    }

    // Spring MVC calls method loadPetWithVisit(...) before processNewVisitForm is called
    @RequestMapping(value = "/owners/{ownerId}/pets/{petId}/visits/new", method = RequestMethod.POST)
    public String processNewVisitForm(@Valid Visit visit, BindingResult result) {
        if (result.hasErrors()) {
            return "pets/createOrUpdateVisitForm";
        } else {
            this.clinicService.saveVisit(visit);
            return "redirect:/owners/{ownerId}";
        }
    }
```

```java
/**
 * Mostly used as a facade so all controllers have a single point of entry
 *
 * @author Michael Isvy
 */
public interface ClinicService {

    Collection<PetType> findPetTypes() throws DataAccessException;

    Owner findOwnerById(int id) throws DataAccessException;

    Pet findPetById(int id) throws DataAccessException;

    void savePet(Pet pet) throws DataAccessException;

    void saveVisit(Visit visit) throws DataAccessException;

    Collection<Vet> findVets() throws DataAccessException;

    void saveOwner(Owner owner) throws DataAccessException;

    Collection<Owner> findOwnerByLastName(String lastName) throws DataAccessException;

}
```

VisitController

VisitService

PetRepository

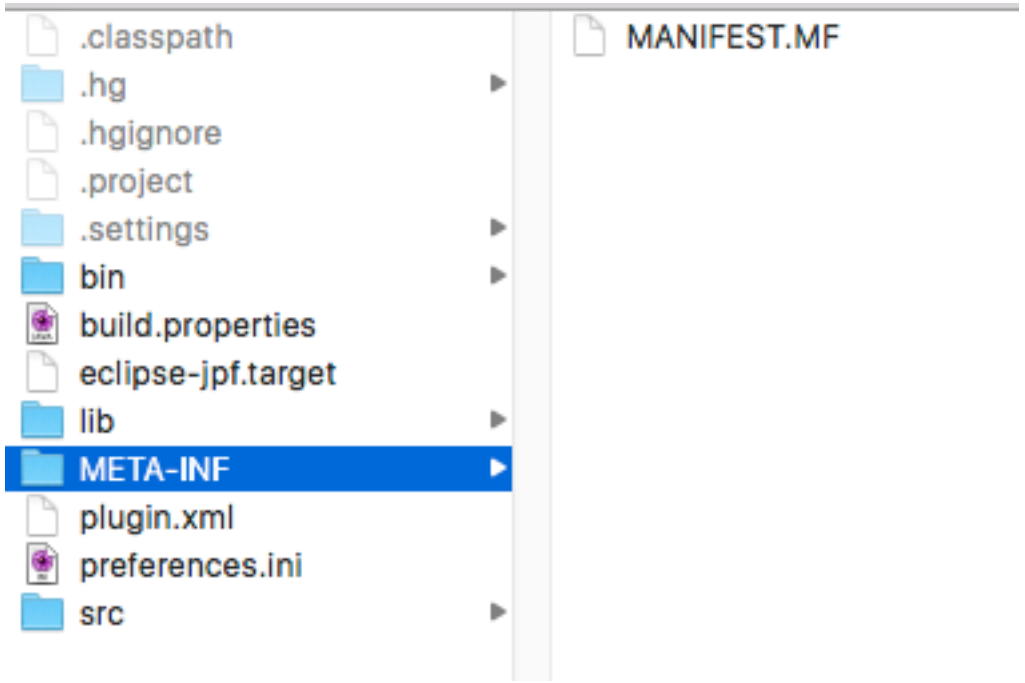
OwnerRepositor

Analysis

- Testability?
- Scalability?
- Flexibility?
- Ease of development?
- Ease of deployment?

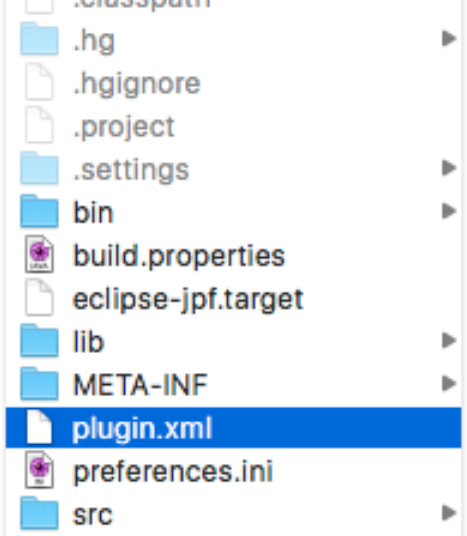
Microkernel architecture

- Eclipse - Equinox - OSGi
- (Atom - Node)



Manifest


```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Eclipse-jpf
Bundle-SymbolicName: eclipse-jpf;singleton:=true
Bundle-Version: 1.0.0.qualifier
Bundle-ClassPath: lib/antlr-runtime-3.4.jar,
 lib/ST-4.0.4.jar,
 lib/jpf-template.jar,
 lib/RunJPF.jar,
 .
Bundle-Activator: eclipse_jpf.Activator
Export-Package: eclipse_jpf,
 gov.nasa.runjpf,
 gov.nasa.runjpf.options,
 gov.nasa.runjpf.util,
 gov.nasa.runjpf.wizard
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime,
 org.eclipse.ui,
 org.eclipse.core.runtime,
 org.eclipse.ui,
 org.eclipse.core.runtime
Bundle-RequiredExecutionEnvironment: JavaSE-1.8
Bundle-ActivationPolicy: lazy
Import-Package: org.eclipse.core.filesystem,
 org.eclipse.core.resources,
 org.eclipse.core.runtime;version="3.4.0",
 org.eclipse.core.runtime.jobs,
 org.eclipse.core.runtime.preferences;version="3.3.0",
 org.eclipse.jdt.core,
 org.eclipse.jface.action,
 org.eclipse.jface.dialogs,
 org.eclipse.jface.preference,
 org.eclipse.jface.text,
 org.eclipse.jface.viewers,
 org.eclipse.jface.wizard,
 org.eclipse.osgi.util;version="1.1.0",
 org.eclipse.swt,
 org.eclipse.swt.events,
 org.eclipse.swt.graphics,
 org.eclipse.swt.layout,
 org.eclipse.swt.widgets,
 org.eclipse.ui,
 org.eclipse.ui.console,
 org.eclipse.ui.dialogs,
 org.eclipse.ui.ide,
```



Extension points

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension
         point="org.eclipse.core.contenttype.contentTypes">
      <content-type
            base-type="org.eclipse.core.runtime.properties"
            file-extensions="jpf"
            id="com.javapathfinder.vjp.configContentType"
            name="JPF Config File"
            priority="normal">
      </content-type>
   </extension>
   <extension
         point="org.eclipse.ui.popupMenus">
      <objectContribution
            adaptable="false"
            id="runjpf.jpfConfigLaunch"
            nameFilter="*.jpf"
            objectClass="org.eclipse.core.resources.IFile">
         <action
               class="gov.nasa.runjpf.VerifyActionDelegate"
               enablesFor="1"
               id="RunJPF.verifyaction"
               label="Verify..."
               menubarPath="additions"
               tooltip="Execute this property file as a JPF configuration">
         </action>
      </objectContribution>
   </extension>
   <extension
         point="org.eclipse.ui.preferencePages">
      <page
            class="gov.nasa.runjpf.options.Preferences"
            id="eclipse-jpf.preferences"
            name="JPF Preferences">
      </page>
   </extension>
      <extension
            point="org.eclipse.core.runtime.preferences">
            <initializer class="gov.nasa.runjpf.options.DefaultPreferences" />
      </extension>
<extension
      point="org.eclipse.ui.newWizards">
   <category
         id="eclipse-jpf.category.wizards"
```

Lifecycle

```
public class EclipseJPF extends AbstractUIPlugin {

    /**
     *  The plug-in ID
     */
    public static final String PLUGIN_ID = "RunJPF";

    // The shared instance
    private static EclipseJPF plugin;

    public EclipseJPF() {
    }

    /*
     * (non-Javadoc)
     * @see org.eclipse.ui.plugin.AbstractUIPlugin#start(org.osgi.framework.BundleContext)
     */
    public void start(BundleContext context) throws Exception {
        super.start(context);
        plugin = this;
    }

    /*
     * (non-Javadoc)
     * @see org.eclipse.ui.plugin.AbstractUIPlugin#stop(org.osgi.framework.BundleContext)
     */
    public void stop(BundleContext context) throws Exception {
        plugin = null;
        super.stop(context);
    }
```

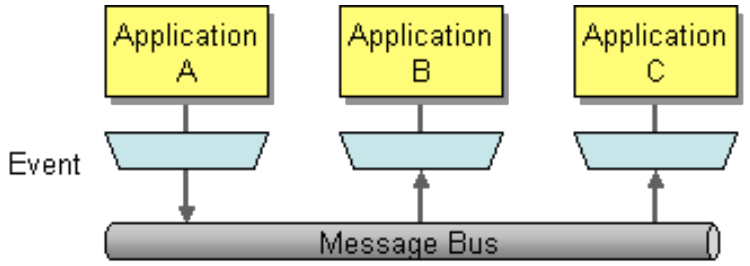
Plugin

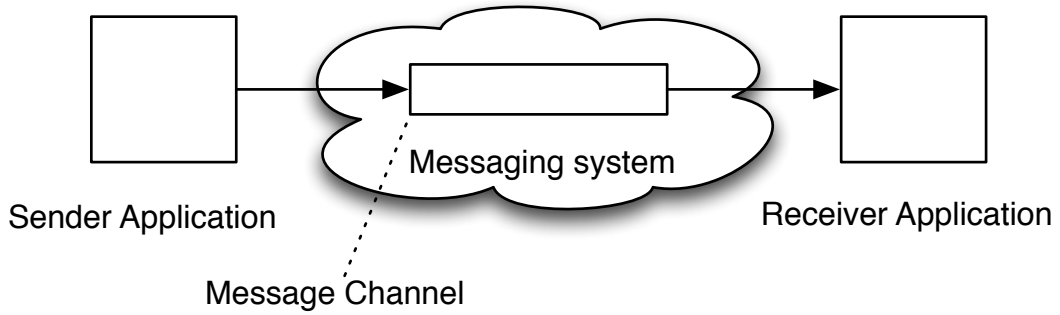
```
public class DefaultPreferences extends AbstractPreferenceInitializer {

  @Override
  public void initializeDefaultPreferences(){
    IPreferenceStore store = EclipseJPF.getDefault().getPreferenceStore();
    store.setDefault(EclipseJPFLauncher.SITE_PROPERTIES_PATH,
EclipseJPFLauncher.DEFAULT_SITE_PROPERTIES_PATH);
    store.setDefault(EclipseJPFLauncher.PORT, EclipseJPFLauncher.DEFAULT_PORT);
  }
}
```

Analysis

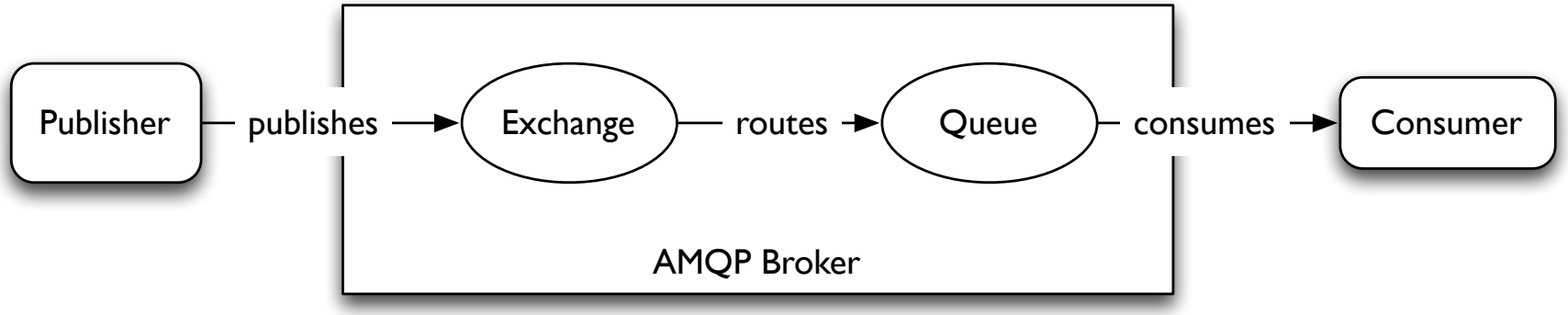
- Testability?
- Scalability?
- Flexibility?
- Ease of development?
- Ease of deployment?

Message bus architectures

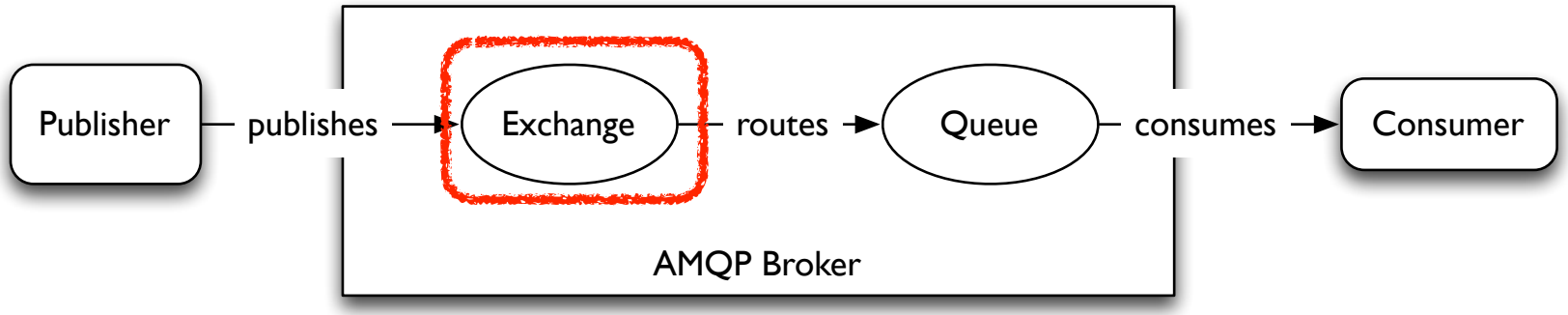


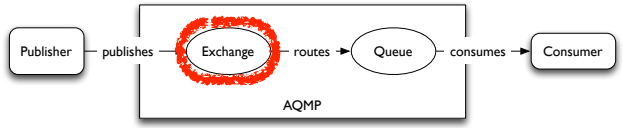


AMQP: Advanced Message Queueing Protocol



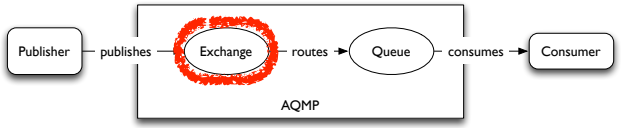
https://www.rabbitmq.com/tutorials/amqp-concepts.html


Publisher
publishes
Exchange
routes
Queue
consumes
Consumer
AMQP Broker

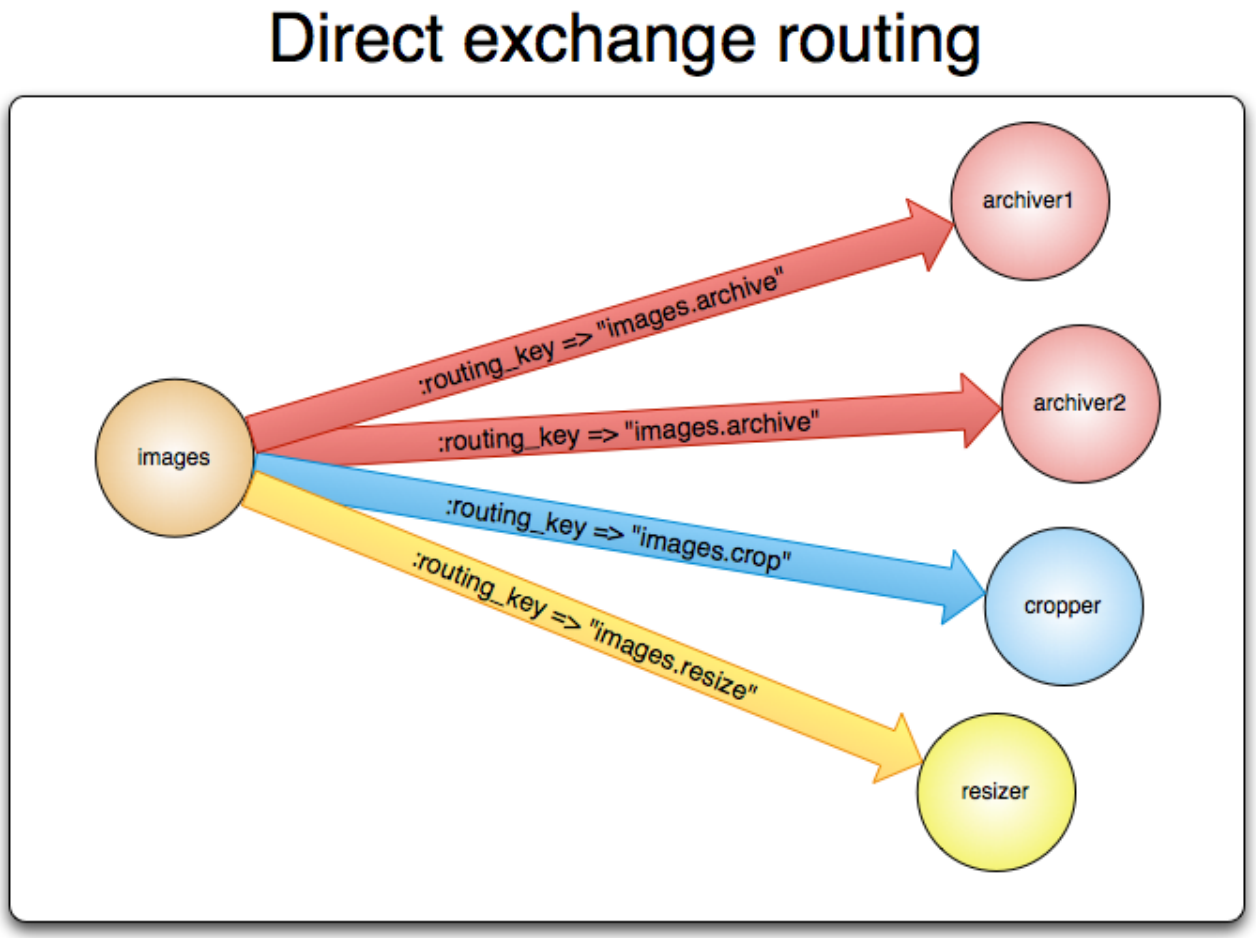

Publisher
publishes
Exchange
routes
Queue
consumes
Consumer
AQMP


Publisher / Consumer
exchange.declare
name = "my.exchange", type = "direct", durable = true
Client
AMQP broker

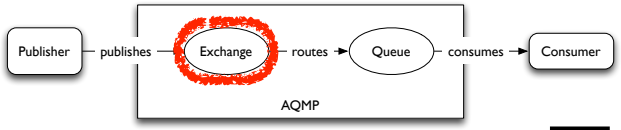

Publisher / Consumer
exchange.declare-ok
Client
AMQP broker



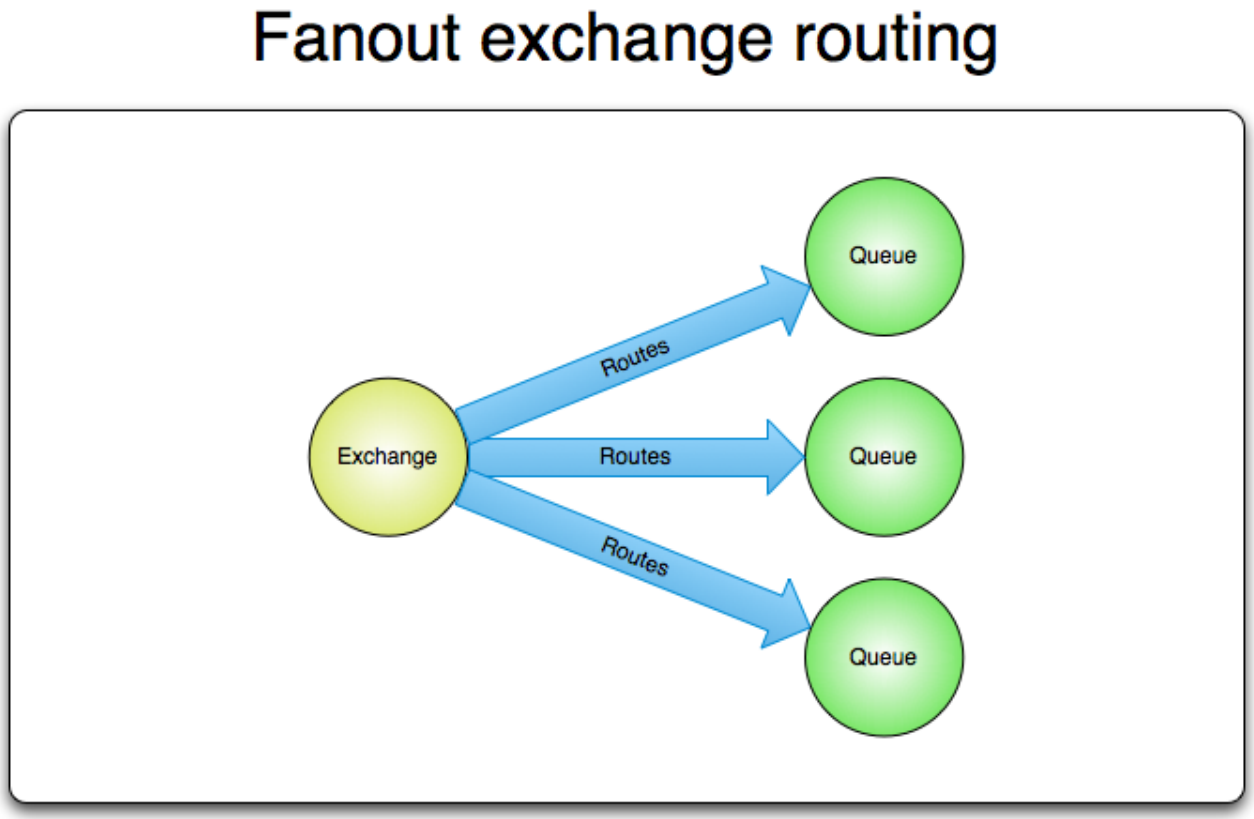
Exchange



~ UDP Unicast



Exchange (2)



~ UDP Broadcast

Analysis

- Testability?
- Scalability?
- Flexibility?
- Ease of development?
- Ease of deployment?

Summary

- Communication, contributions
- Coding conventions - design principles - high-level structures

Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files … — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools

Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files … — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools

Version control - GIT

Sample Disorganized Project

- “Hey, Anders, could you send me a copy of those changes you made last Tuesday?”
- “Mikael, this function doesn’t work anymore. Did you change something?”
- “Sorry, I can’t seem to find those old classes. I guess you’ll just have to re-implement them.”
- “OK, we’ve all been working hard for the last week. Now let’s integrate everyone’s work together.”

What is version control?

Basic functionality:

- keep track of changes made to files (allows roll-backs)
- merge the contributions of multiple developers

Benefits:

- facilitates backups
- increased productivity (vs manual version control)
- encourages experimentation
- helps to identify/fix conflicts
- makes source readily available – less duplicated effort

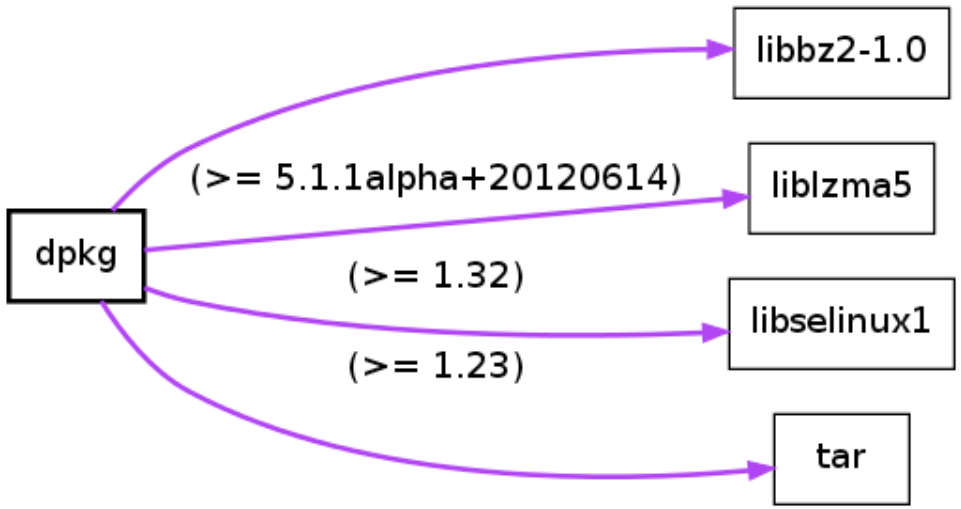
Additional benefits

- Accountability
  - who wrote the code?
  - do we have the rights to it?
- Support software engineering
  - hooks for peer reviews
- Software branches
  - different versions of software need to be maintained, ensure bug fixes shared
- Record Keeping
  - Commit logs may tie to issue tracking system or be used to enforce guidelines

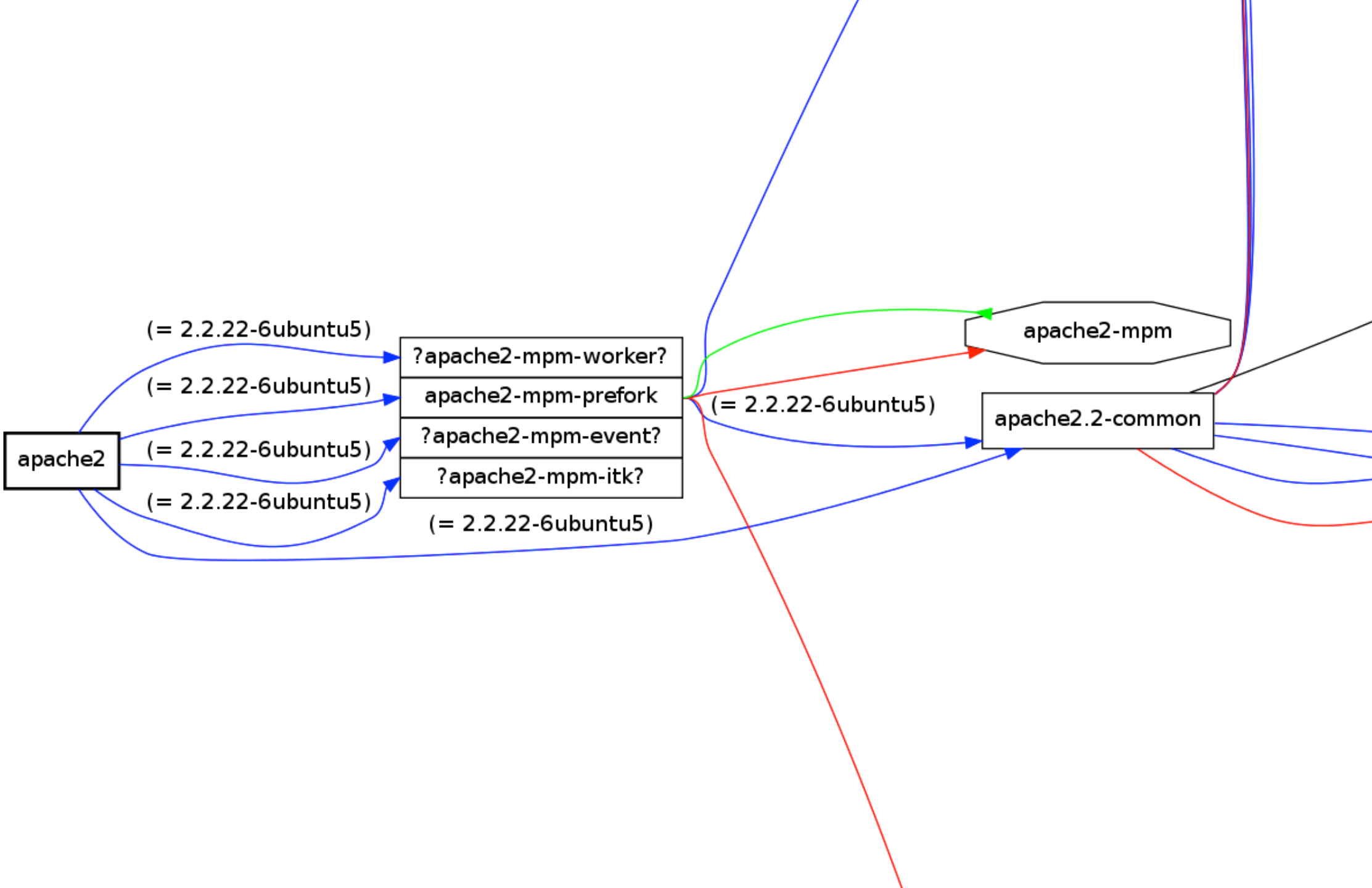
Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files … — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools

Package management systems



Dep


apache2
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
(= 2.2.22-6ubuntu5)
?apache2-mpm-worker?
apache2-mpm-prefork
?apache2-mpm-event?
?apache2-mpm-itk?
(= 2.2.22-6ubuntu5)
apache2-mpm
apache2.2-common

Dependency management issues

- Is a request to modify the current software component graph satisfiable?
  - Are additions compatible with other components?
  - Are deletions safe with respect to other dependencies?
- Given a component, determine versions of other components we can safely rely on

Dependency management as satisfiability

```
(a | b | c) & (d | e | f) ... = TRUE

(a | b | c) & (-c) & (-b | -a) ... = TRUE
```

Rules

A requires B provided by B1, B2, B3
Rule: (-A | B1 | B2 | B3)

A conflicts with B provided by B1, B2, B3
3 Rules: (-A | -B1), (-A | -B2), (-A | -B3)

Dependency management issues

- Y depends on X >= 1.8. X makes binary incompatible changes from v. 1.9 to v. 2.0…
- Can components be installed from local sources as well as from remote?
- Should OS-specific dependency management or language-specific be used?

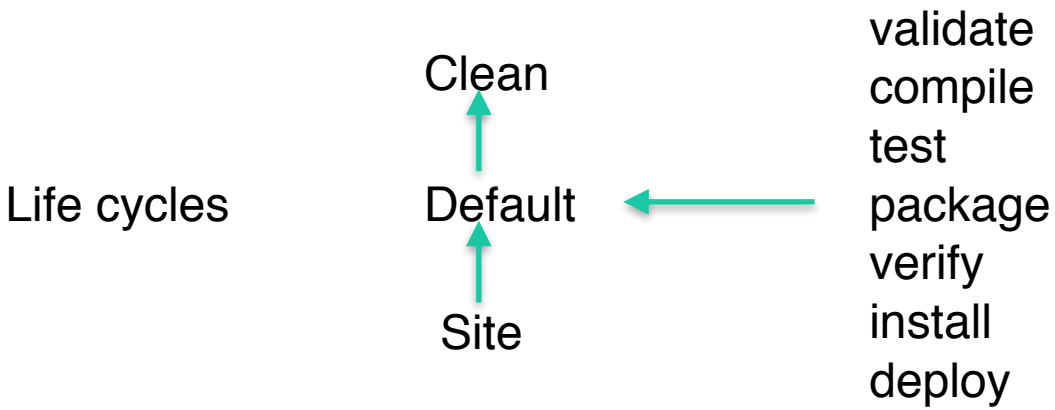
Software package management systems

Name	Environment	Format
NuGet	.Net CLR	XML
Gradle	JVM	XML
dpkg/APT	Linux	Ar archive
Rubygems	Ruby	Ruby
MSI	Windows	In-file DB
BSD Ports	OS X/Linux/BSD	Makefile
...		

Maven


Maven™

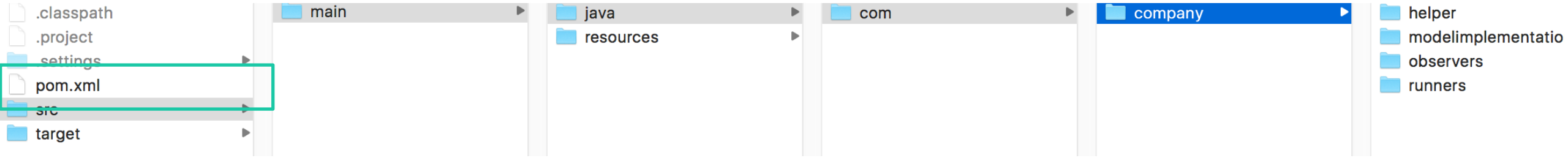
```
mvn -h
```


Life cycles
Clean
Default
Site
validate
compile
test
package
verify
install
deploy

Maven example


```
Olas-MacBook-Pro:java-petclinic olale$ mvn graphwalker:test
[INFO] Scanning for projects...
[ … ]
[INFO] >>> graphwalker-maven-plugin:3.4.2:test (default-cli) > [graphwalker]test-compile @ java-petclinic >>>
[INFO]
[INFO] --- graphwalker-maven-plugin:3.4.2:generate-sources (generate-sources) @ java-petclinic ---
[INFO]
[INFO] --- graphwalker-maven-plugin:3.4.2:validate-models (default-cli) @ java-petclinic ---
[INFO]
[INFO] --- graphwalker-maven-plugin:3.4.2:generate-sources (default-cli) @ java-petclinic ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ java-petclinic ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 10 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ java-petclinic ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- graphwalker-maven-plugin:3.4.2:validate-test-models (default-cli) @ java-petclinic ---
```

Maven – structure




```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Maven – Configuration

```
<parent>
    <groupId>org.graphwalker.example</groupId>
    <artifactId>graphwalker-example</artifactId>
    <version>3.4.2</version>
</parent>

<artifactId>java-petclinic</artifactId>
```

```
<build>
      <plugins>
          <plugin>
              <groupId>org.apache.maven.plugins</groupId>
              <artifactId>maven-compiler-plugin</artifactId>
              <version>3.1</version>
              <configuration>
                  <source>1.7</source>
                  <target>1.7</target>
              </configuration>
          </plugin>
          <plugin>
              <groupId>org.graphwalker</groupId>
              <artifactId>graphwalker-maven-plugin</artifactId>
              <version>${project.version}</version>
              <!-- Bind goals to the default lifecycle -->
              <executions>
                  <execution>
                      <id>generate-sources</id>
                      <phase>generate-sources</phase>
                      <goals>
                          <goal>generate-sources</goal>
                      </goals>
                  </execution>
              </executions>
          </plugin>
      </plugins>
</build>
```

Scenarios - actions

1. Single developer, multiple changes — Version control system
2. Many developers, multiple changes — Distributed version control system
3. Many groups of developers, multiple changes — Package management system
4. Building artefacts based on multiple files with dependencies — build scripts
5. Conducting multiple actions with inter-dependencies on multiple files … — Flexible build system
6. Automatically sensing changes and conducting such actions based on changes — Continuous integration tools

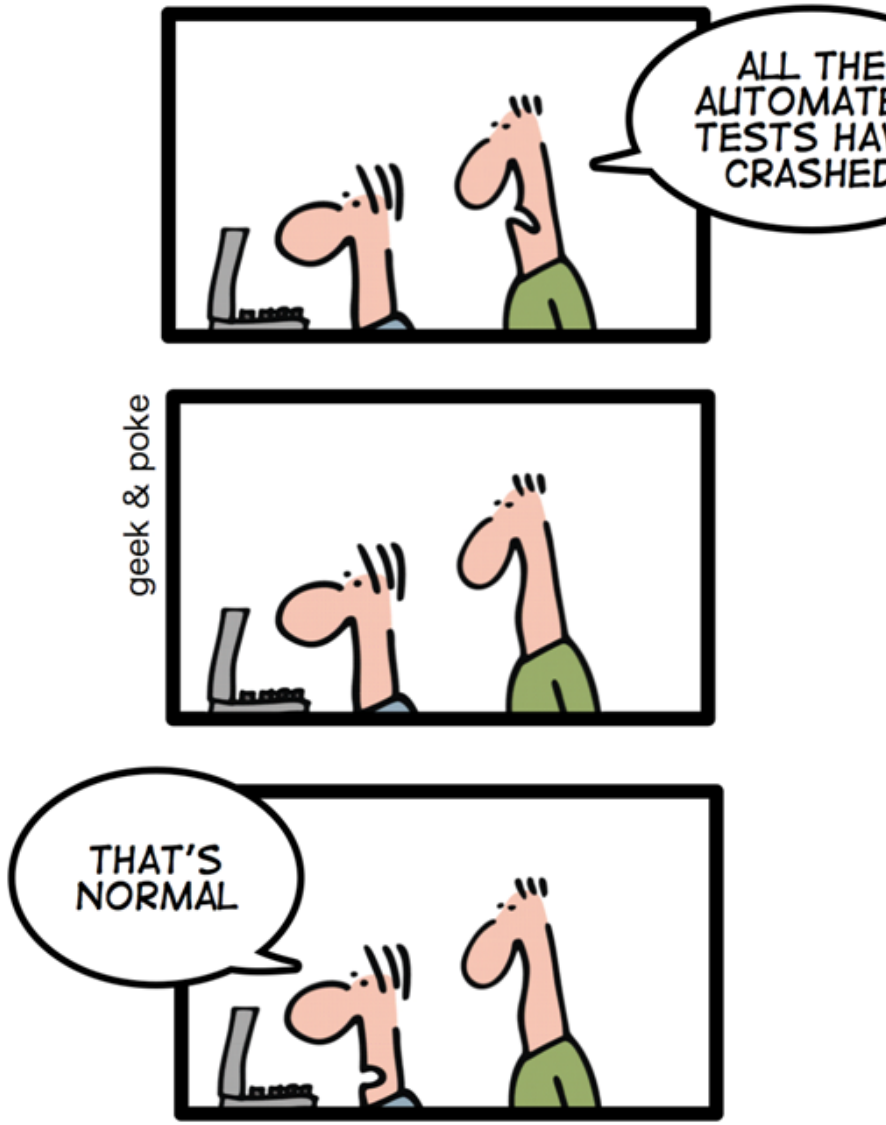
CI - Continuous Integration

“Continuous Integration is 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. “

Martin Fowler

Geek & Poke List of Best Practises

CI Gives you the comforting feeling to know that everything is normal



https://www.zuehlke.com/blog/app/uploads/2015/11/geek-and-poke.png

Why?

Detect development problems earlier

Reduce risks of cost, schedule and budget

Find and remove bugs earlier

Deliver new features and get user feedback more rapidly

How?

Maintain a single source repository

Automate the build

Make your build self-testing

Keep the build fast

Keep the build on the CI machine

Test in a clone of production environment

Make it easy for everyone to get the latest executable

Make the process transparent for everyone

CI and CD

Summary

Continuous Integration

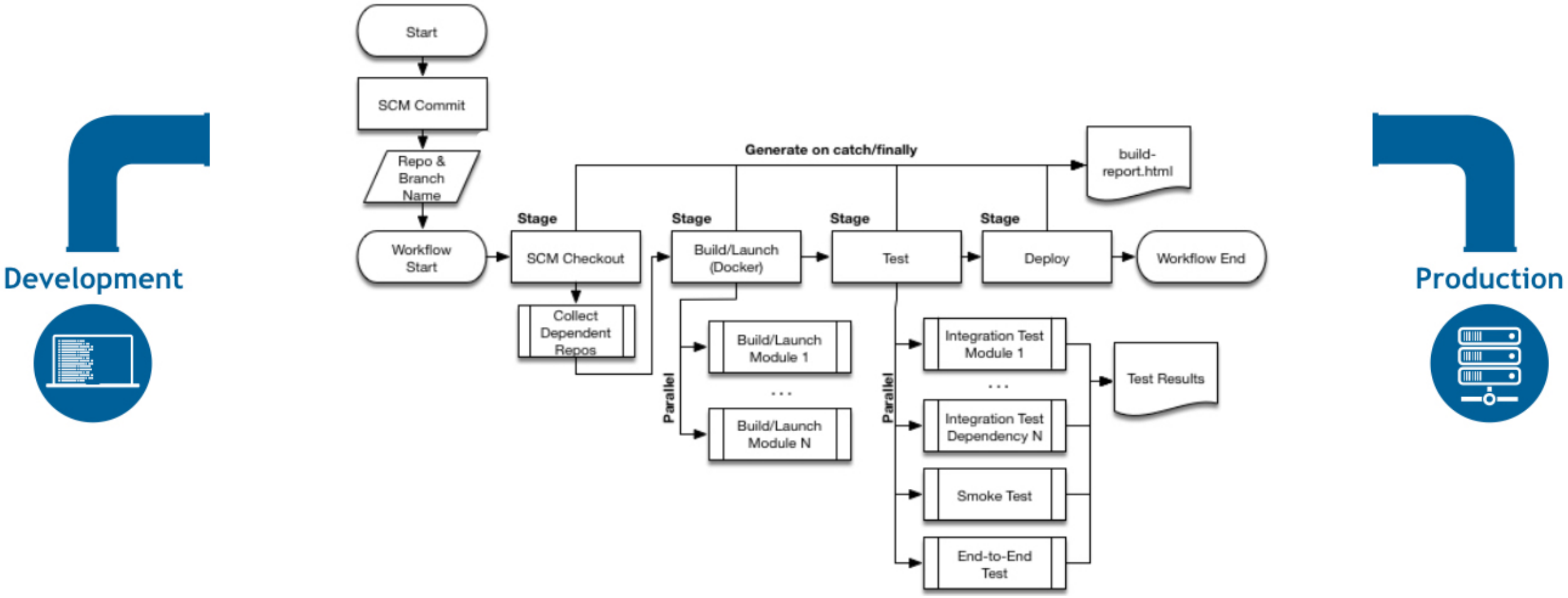
Continuous Delivery

Continuous Deployment

Jenkins

Jenkins

Workflow automation tool



Jenkins

Workflow automation tool - pipelines

```
node { // <1>
    stage('Build') { // <2>
        sh 'make' // <3>
    }

    stage('Test') {
        sh 'make check'
        junit 'reports/**/*.xml' // <4>
    }

    stage('Deploy') {
        sh 'make publish'
    }
}
```

Groovy (JVM-based language)

Travis CI

A pull request is created

GitHub tells Travis CI the build is mergeable

Hooray! Your build passes!

Travis CI updates the PR that it passed

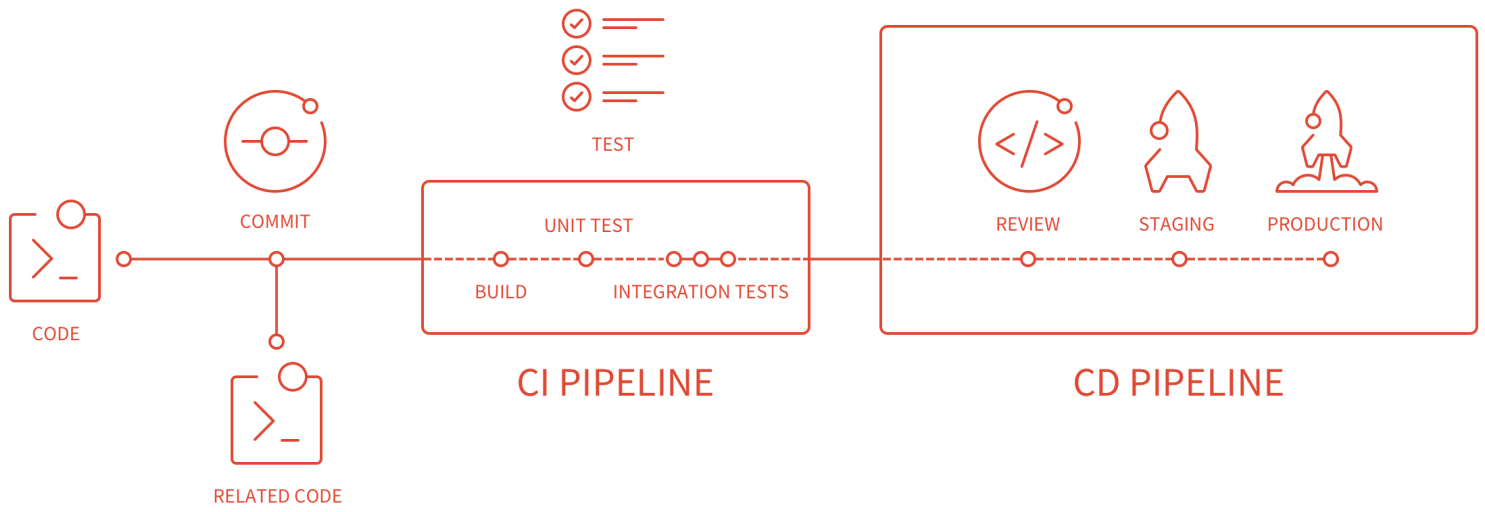
You merge in the PR goodness

GITLAB CI

Code and build scripts in the same repo
Easy to start
Scalable
Isolated test environment

Gitlab CI: Pipelines and Stages

A pipeline is a group of jobs that get executed in stages(batches). All of the jobs in a stage are executed in parallel, and if they all succeed, the pipeline moves on to the next stage. If one of the jobs fails, the next stage is not executed.
Pipelines are defined in .gitlab-ci.yml by specifying jobs in stages:



```
image: docker:latest
services:
  - docker:dind

variables:
  DOCKER_DRIVER: overlay
  SPRING_PROFILES_ACTIVE: gitlab-ci

stages:
  - build
  - package
  - deploy

maven-build:
  image: maven:3-jdk-8
  stage: build
  script: "mvn package -B"
  artifacts:
    paths:
      - target/*.jar

docker-build:
  stage: package
  script:
  - docker build -t registry.gitlab.com/marcolenzo/actuator-sample .
  - docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN registry.gitlab.com
  - docker push registry.gitlab.com/marcolenzo/actuator-sample

k8s-deploy:
  image: google/cloud-sdk
  stage: deploy
  script:
  - echo "$GOOGLE_KEY" > key.json
  - gcloud auth activate-service-account --key-file key.json
  - gcloud config set compute/zone europe-west1-c
  - gcloud config set project actuator-sample
  - gcloud config set container/use_client_certificate True
  - gcloud container clusters get-credentials actuator-sample
  - kubectl delete secret registry.gitlab.com
  - kubectl create secret docker-registry email=lenzo.marco@gmail.com
  - kubectl apply -f deployment.yml
```