Deploying Services

Deployment Maths

~ 70 software components
+ ~ 12 computers
+ Different configurations for each

= Headache

Approach

- Need to componentise capabilities
- Need to support component inter-dependencies and versioning
- Control deployment and configuration in one place

Componentisation – OSGi

- OSGi is a runtime for software *bundles*
- Bundles can be libraries or active processes
- OSGi manages dependencies and versioning
- Mature and well supported
  - Three open source implementations
  - Foundation of Eclipse 3.0

Deployment – OSGi Client

- Core of solution: OSGi component model
- Clients boot an OSGi shell which...
    - ... loads OSGi software bundles from server which...
    - ...read their configuration from the server
- All software and configuration from server
- All software packaged as OSGi bundles

Configuration – Livespace OSGi Services

- The Livespace deployment and configuration system are themselves OSGi bundles
- Livespace services bundles are
  - activated by the deployment bundle
  - can load configuration from the configuration bundle

The Livespace Client

- A tiny (6K) JAR that pulls across OSGi and the Livespace bootstrap bundle via HTTP from a server
- Starts OSGi + bootstrap bundle
- Bootstrap takes over from there

The Livespace Server

- Same OSGi environment as client
- Bundles loaded from local file-system rather than HTTP
- Provides bundles and configuration to clients via HTTP server
- Runs most of the entity-level room services: clipboard, meta app repository, ... etc

Developing A Service

- A new service is typically developed outside of OSGi
- Service usually has no dependency on OSGi
- Once developed, add OSGi wrapping

Bundling A Service

1. Write an activator (like `main ()`)
2. Write an Ant build target to create bundle
3. Deploy
4. Fix inevitable dependency snafu’s

Bundling A Service – The Activator

- Loads configuration (if any)
- Resolves dependencies on other services
- Creates and starts service instance

Bundling A Service – Simple Activator

```
public class Activator extends RoomBasedServiceActivator
{
  protected ILivespaceService createService ()
  {
    return new ClipboardService (elvin, room);
  }
}
```

Bundling A Service – Complex Activator

```
public class Activator implements BundleActivator
{
  private Dependencies dependencies;
  private CommandLineService service;

  public void start (BundleContext context)
  {
    dependencies = new Dependencies ();

    dependencies.add
      ("computer",
       new ServiceDependency (context, IComputerService.class));

    dependencies.readyListeners.add
      (new Delegate (this, "startService", context));
    dependencies.notReadyListeners.add
      (new Delegate (this, "stopService"));

    dependencies.start ();
  }

...
```

Bundling A Service – Complex Activator

```
protected void startService (BundleContext context)
{
  IComputerService computerService =
    (IComputerService)dependencies.get ("computer");

  service =
    new CommandLineService
      (computerService.getComputer (),
       Configuration.configurationFor ("livespace.command_line"));
}

protected void stopService ()
  throws Exception
{
  service.stop ();
  service = null;
}
```

Bundling A Service – The Bundle Target

```
<target name="bundle_server" depends="compile">
  <jar jarfile="livespace.services.clipboard-1.0.0.jar">

    <manifest>
      <attribute name="Bundle-Name" value="livespace.services.clipboard" />
      <attribute name="Bundle-Version" value="1.0.0"/>
      <attribute name="Bundle-Description" value="Livespace shared clipboard service"/>
      <attribute name="Bundle-Activator
                 value="livespace.services.clipboard.Activator" />
      <attribute name="Bundle-Classpath" value="." />
      <attribute name="Bundle-Vendor" value="DSTO" />
      <attribute name="Bundle-Category" value="service" />
      <attribute name="Import-Package"
                value="dsto.dfc.databeans,dsto.dfc.logging,livespace.services,
                       livespace.services.entities,livespace.services.room,
                       livespace.osgi,org.elvin.je4,
                      org.osgi.framework,org.osgi.service.cm" />
    </manifest>

    <fileset dir="classes" includes="livespace/services/clipboard/*.class"/>
  </jar>
</target>
```

Deploying Bundles

- How to control which bundles are deployed to which hosts
- How to specify settings for services in those bundles

Bundle Deployment Sets

- Deployment specified in *deploy set* files
- Bundles can be targeted:
  - Globally – deployed to all hosts
  - By category – e.g. deployed to “server” hosts
  - By host name
- Deploy sets are merged in the above order

Bundle Run Levels

- Each bundle has a *run level*
- Same concept as UNIX run levels
- level < 5: system bundles
- level = 5: headless services
- level = 6: interactive bundles (UI)

Service Configuration

- Services configured using Java property files
- Same global/category/host merging scheme as used for deploy sets

Service Configuration

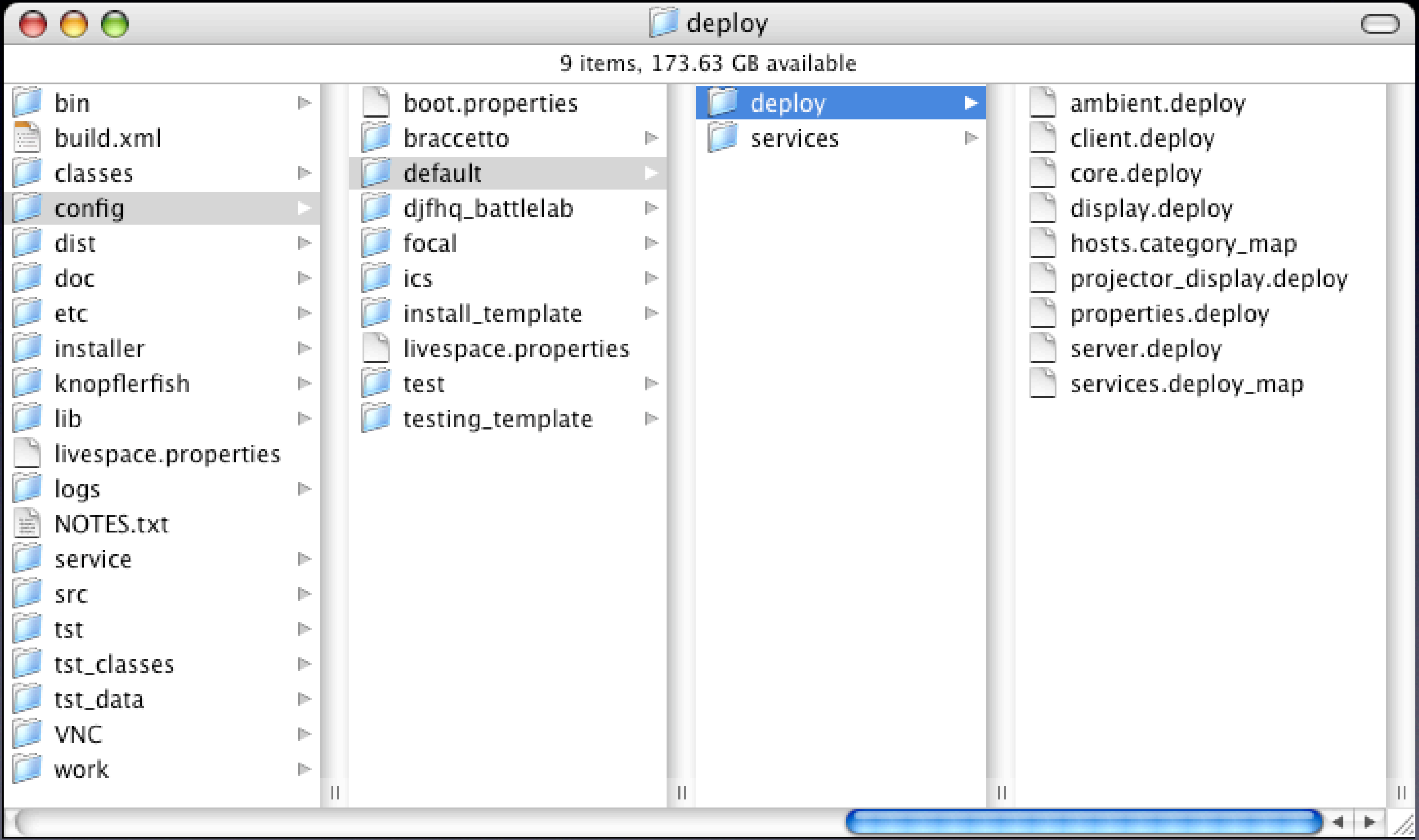
config/default/services/livespace.room.properties

```
# The name of the room
name=${room_name}

createService=false

datastore=file://${main_server}/datastore

mediaBaseURL=http://${main_server}:8090/livespaces/media
```

Configuration Layout

- `$LIVESPACE_OSGI_PATH/config` points to root of config area
  - e.g. `http://main_server:8090/config`
- Configuration specified by merging files in two directories:
  - *default* – preset settings
  - *current* – customisations for local room

Basic Deployment



deploy

9 items, 173.63 GB available

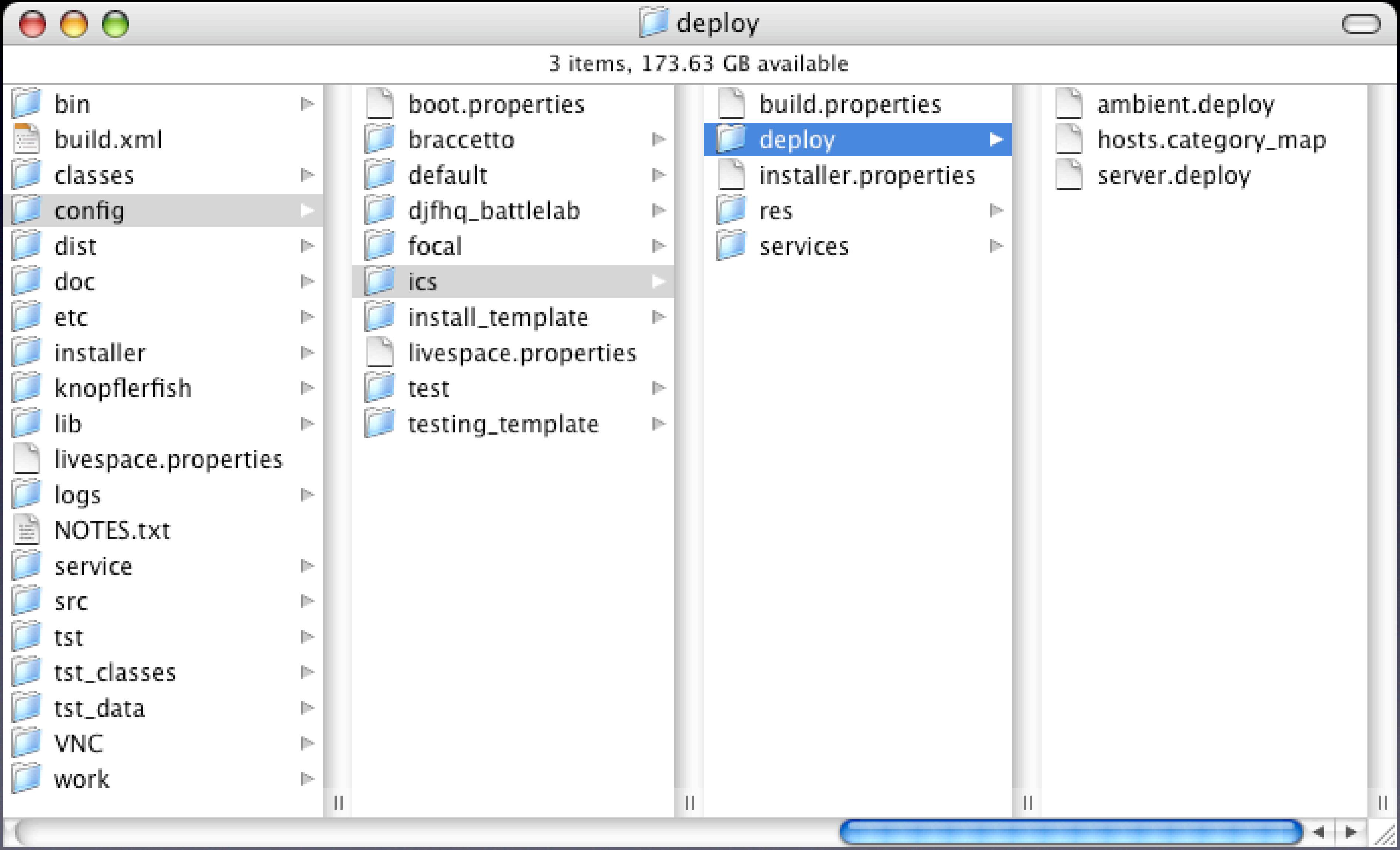
bin
build.xml
classes
config
dist
doc
etc
installer
knopflerfish
lib
livespace.properties
logs
NOTES.txt
service
src
tst
tst_classes
tst_data
VNC
work

boot.properties
braccetto
default
djfhq_battlelab
focal
ics
install_template
livespace.properties
test
testing_template

deploy
services

ambient.deploy
client.deploy
core.deploy
display.deploy
hosts.category_map
projector_display.deploy
properties.deploy
server.deploy
services.deploy_map

ICS Deployment



bin
build.xml
classes
config
dist
doc
etc
installer
knopflerfish
lib
livespace.properties
logs
NOTES.txt
service
src
tst
tst_classes
tst_data
VNC
work

boot.properties
braccetto
default
djfhq_battlelab
focal
ics
install_template
livespace.properties
test
testing_template

build.properties
deploy
installer.properties
res
services

ambient.deploy
hosts.category_map
server.deploy

Deploying Bundles

- How do we map deploy sets to hosts?
- `hosts.category_map` – maps from host name to category
- `services.deploy_map` – maps from category/hostname to deploy sets for the host

Category Map

config/ics/deploy/hosts.category_map

```
host.ics-winserver=server

host.ics-display-lft=projector_display
host.ics-display-ctr=projector_display
host.ics-display-rht=projector_display

host.ics-ambient=ambient

host.dent=display
host.beeblebrox=display

host.ics-autm1=client
host.ics-autm2=client
```

Deployment Map

config/default/deploy/services.deploy_map

```
# Default config is client
default=client

# The services on the main server: server setup plus client front end
# You may want to remove the client config for headless servers
category.server=client + server

# clients simply get the client deploy config
category.client=client

# The display servers get client plus display extras
category.display=client + display

category.projector_display=client + projector_display

# The ambient display server
category.ambient=client + ambient
```

Deploy Sets

config/default/deploy/core.deploy

```
initlevel 2

# Basic OSGi bundles
start util-1.0.0
start cm_all-1.0.1
start log_all-1.0.1
install jsdk-2.2
start http_all-1.1.0

initlevel 4

# Livespace core bundles
install livespace.common
install livespace.osgi
install livespace.services.osgi

start livespace.elvin
start livespace.osgi.controller
start livespace.logging.remote

initlevel 5

# Livespace services
start livespace.services
start livespace.services.room
start livespace.services.computer
start livespace.services.osgi_admin
start livespace.http
```

config/default/deploy/server.deploy

```
include core

initlevel 4

# Server hosts central log console
start livespace.logging.console

# Core room services

initlevel 5

start livespace.services.computer.guests
start livespace.services.clipboard
start livespace.services.room_presentation
start livespace.services.sessions
start livespace.services.meta_apps
start livespace.services.screen_sharing.server
start livespace.services.teamthink
```

Exercise 2 – Deploying The Sound Player