

LCFG

**A practical tool for
system configuration**

Paul Anderson

dcspaul@ed.ac.uk



[http://homepages.inf.ed.ac.uk/dcspaul/
publications/oslo-2008b-talk.pdf](http://homepages.inf.ed.ac.uk/dcspaul/publications/oslo-2008b-talk.pdf)

LCFG

- What is LCFG ?
- How does it work ?
 - *the components*
 - *the client*
 - *the server*
- Why does it work this way ?
- Managing specifications
- Some open issues

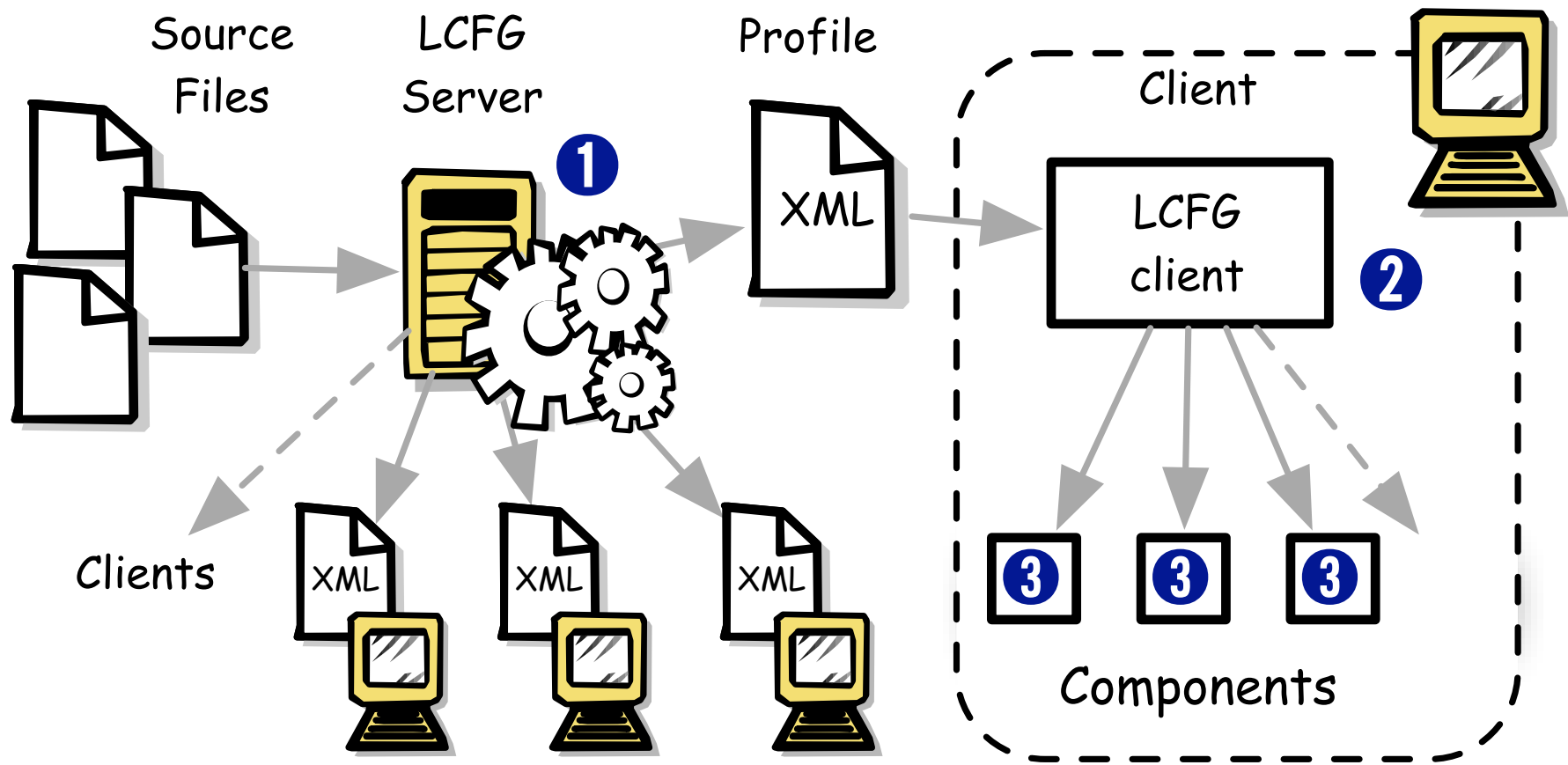


LARGE SCALE UNIX CONFIGURATION SYSTEM

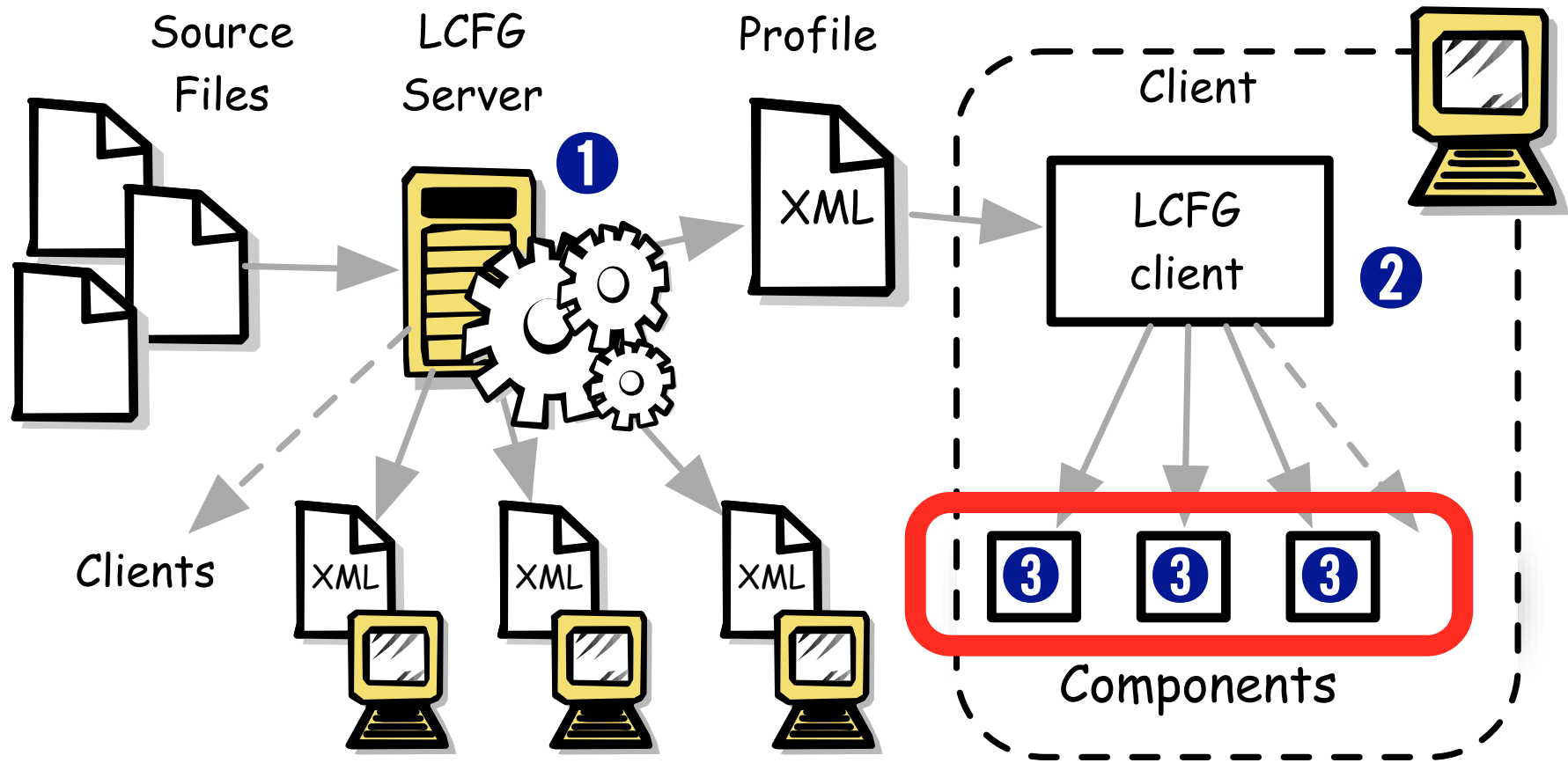
<http://www.lcfg.org>

- Developed around 1994 in Computer Science at Edinburgh University
 - *under regular development since*
 - *used heavily at Edinburgh for production systems*
 - *the basis for research projects*
- Distributed under the GPL
 - *but not widely adopted elsewhere*
- Currently very stable and being promoted more

How does LCFG work ?



LCFG components



LCFG components

- An LCFG **component** is a script which manages some particular “subsystem” (eg. mail)
- It takes a set of parameters called **resources** and configures the subsystem accordingly
 - *compute file contents from the resources*
 - *create/modify configuration files*
 - *restart daemons when resources change*
- Different versions of a component may cater to different circumstances
 - *simple/complex versions of the DNS component*
- Components are independent

Abstracting the details

```
snmpd : localhost 129.215.216. 129.215.58.
```

```
stunnel : localhost .inf.ed.ac.uk .dcs.ed.ac.uk
```

```
slapd : 127.0.0.1 R$* < @ [ IPv6 : $+ ] > $*    $: $2 $| $1 < @@ [ $(dequote $2 $) ] > $3  
mark IPv6 addr
```

```
R$+ $| $* < @@ $=w > $*    $: $2 < @ $j . > $4    self-literal
```

```
M R$+ $| $* < @@ [ $+ ] > $*    @$ $2 < @ [ IPv6 : $1 ] > $4    canon IP addr
```

```
Modeline "1024x768" 65 1024 1032 1176 1344 768 771 777 806 -hsync -  
vsync
```

- 👁 We need to do more than just manipulate configuration files
- 👁 We abstract the **relevant** parts of the contents
- 👁 What is relevant depends on what varies at a particular site

An example

- Assume the `/etc/motd` file varies slightly according to the department
- We might use a template which is identical on every machine

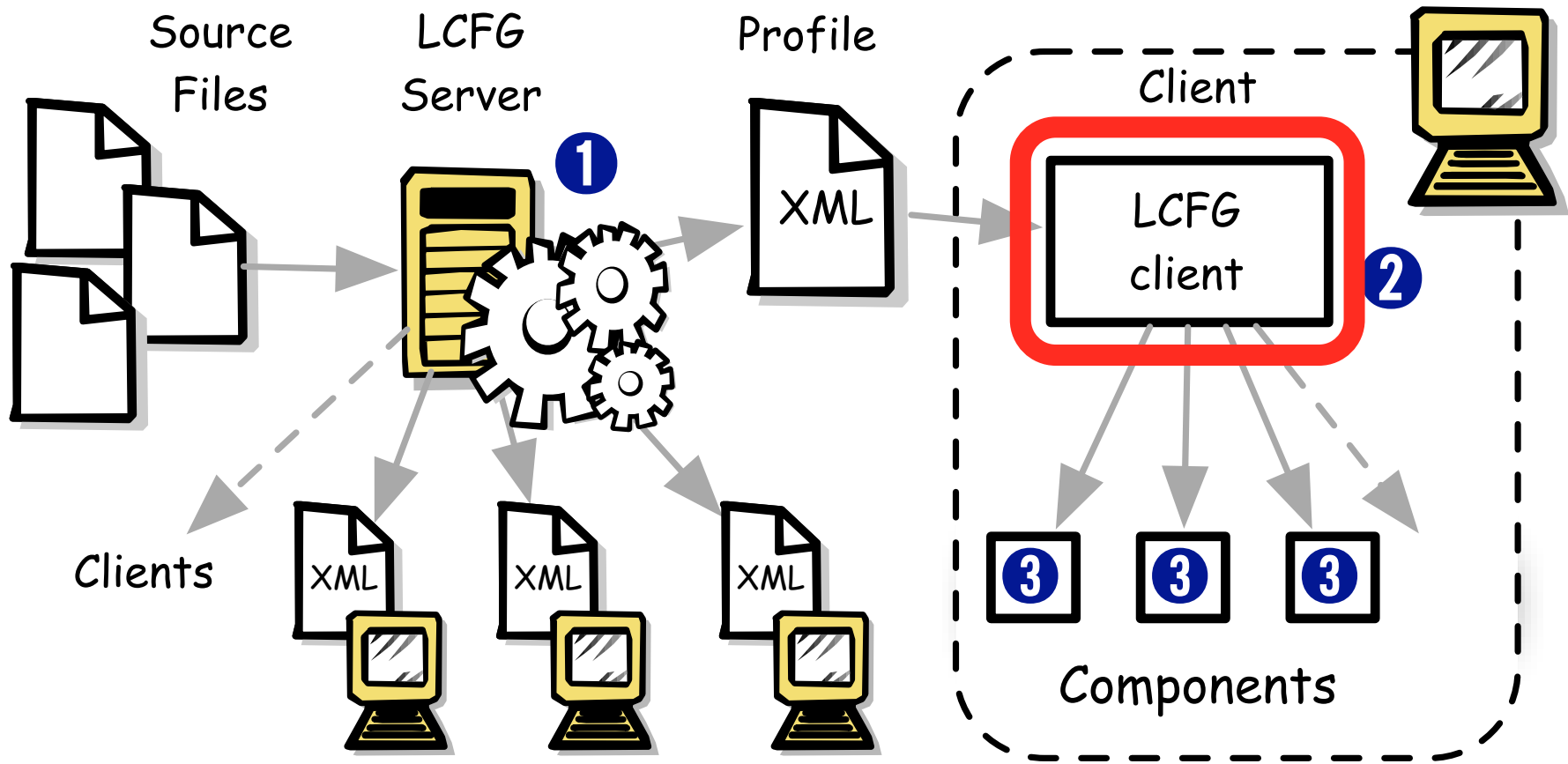
```
Welcome to the University of Foobar  
This is the department of <%motd.dept%>
```

- The profile would contain only the value of the resource `motd.dept`
- The `motd` component would maintain the file using the template and the resource value

Some available components

lcfg-afs, lcfg-alias, lcfg-amd, lcfg-apache, lcfg-arpwatch, lcfg-auth, lcfg-client, lcfg-condor, lcfg-cosign, lcfg-cron, lcfg-cvs, lcfg-cyrussasl, lcfg-dhcpd, lcfg-dns, lcfg-etcservices, lcfg-example, lcfg-file, lcfg-fstab, lcfg-gdm, lcfg-gridengine, lcfg-grub, lcfg-inventory, lcfg-iptables, lcfg-kernel, lcfg-kx509, lcfg-mailcap, lcfg-mailng, lcfg-matlab, lcfg-mozilla, lcfg-mysql, lcfg-network, lcfg-nfs, lcfg-nsswitch, lcfg-ntp, lcfg-openldap, lcfg-openssh, lcfg-openvpn, lcfg-pam, lcfg-perlex, lcfg-postgresql, lcfg-ramdisk, lcfg-rmirror, lcfg-rsync, lcfg-samba, lcfg-server, lcfg-snmp, lcfg-subversion, lcfg-syslog, lcfg-tcpwrappers, lcfg-updaterpms, lcfg-webdav, lcfg-xfree, lcfg-xinetd

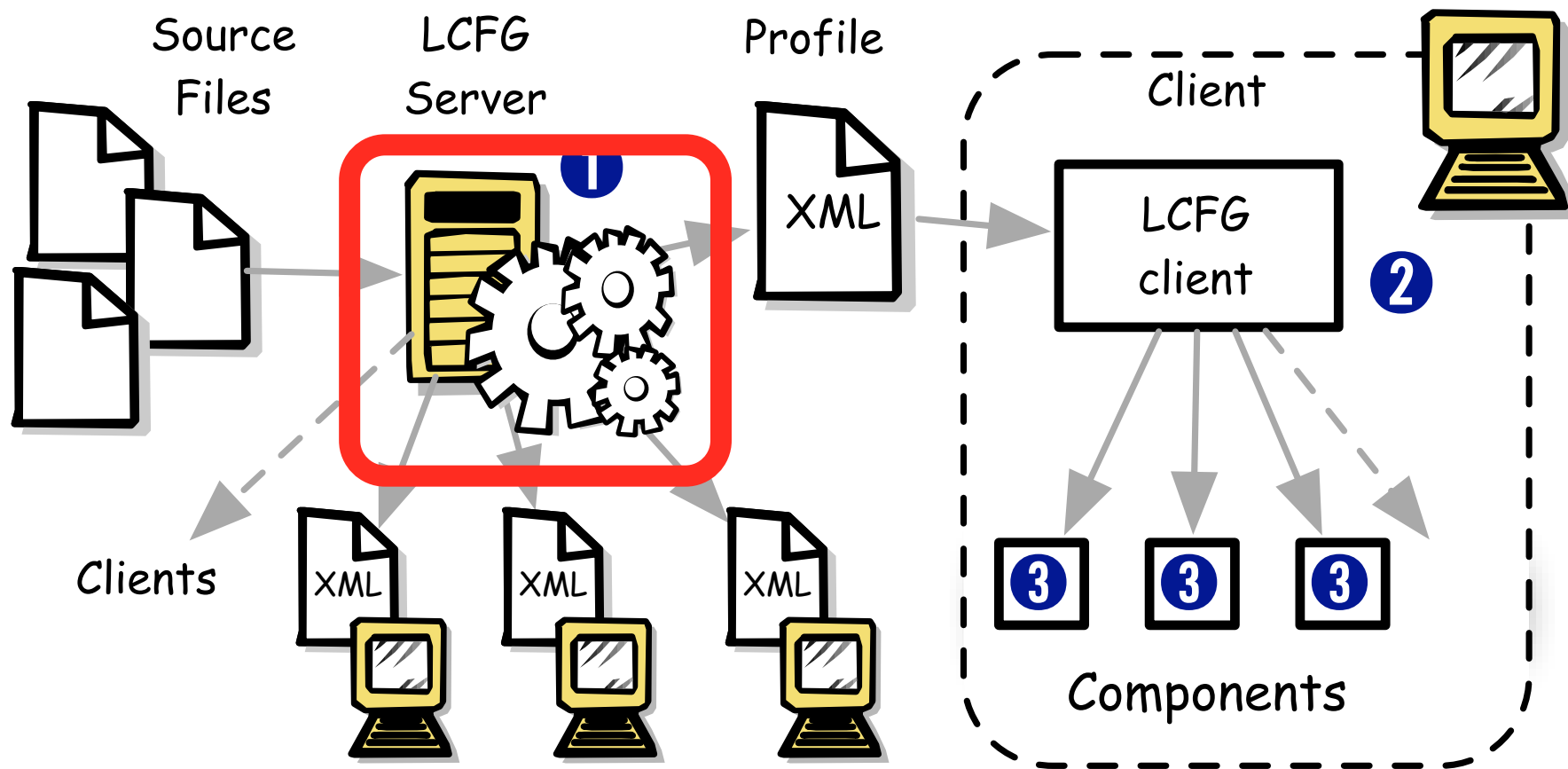
The LCFG client



The LCFG client

- The LCFG client downloads an XML profile from the server whenever the profile changes
- The profile contains all the resources for all the components on the client machine
- The client notifies components whose resources have changed, so that they can reconfigure
- It can return the success of the reconfiguration to the server for monitoring
- The client itself is configured by an LCFG component

The LCFG server



The LCFG server

- The LCFG server generates the profile for each managed machine from a **source** file for the machine
- The source file includes explicit resources specific to the particular machine
- It also includes references to other “aspects”
 - *think “classes” or “header files”*
 - *eg. “web server” or “student lab machine”*
- The server is capable of “composing” resource values from different aspects
- The client is notified when the profile changes

The LCFG source language

- The LCFG source syntax has a long history and can be rather awkward
 - *but the basic facilities have proven themselves*
- Define resources
 - *motd.dept School of Informatics*
- Include aspects
 - *#include “web-server.h”*
- Some other features
 - *references*
 - *“mutation” for aspect composition*
 - *“spanning maps” for cross-machine relationships*

An example source file

```
#include <local/site.h>
#include <lcfg/os/minimal.h>
#include <lcfg/hw/vmware_ws5.h>

!file.files          mADD(example)
file.file_example   /etc/motd
file.type_example   template
file.tmpl_example   /etc/motd.template
!file.variables     mADD(department)

file.v_department   School of Informatics
```

Why does LCFG work like this ?

- Clear declarative specifications
 - *simple, uniform language*
 - *global validation and generation possible*
- Different levels of users
 - *component authors (code)*
 - *senior admins (header files)*
 - *junior admins (source files)*
- Modular “components”
 - *easy for people to write and share*
 - *extensible*
- Support for “prescriptive configuration”

Declarative specifications

- There seems to be a general agreement that configuration tools should be “declarative”
 - *the language specifies the desired configuration*
 - *not the process required to achieve it*
 - *the tool computes the necessary configuration changes*
- Non-declarative tools cause difficulties
 - *all operations must be explicitly idempotent*
 - *we cannot reason about the configuration state*
- Declarative tools must compute the necessary deployment operations

Prescriptive specifications

- A prescriptive tool will specify the complete configuration of the target system
 - *the target system can be recreated completely by the tool*
- Non-prescriptive tools need part of the specification to be provided from elsewhere
 - *manually, or by some other tool*
- Care is needed with non-prescriptive
 - *systems with apparently identical configurations may diverge as parameters are moved in and out of the scope*

Managing specifications

- Composing requirements
 - *from different users*
 - *for different systems*
- Managing relationships
 - *“spanning maps”*
- Change control
- Autonomic configuration
 - *inter-operating with higher-level software*
 - *research links with SmartFrog & PoDIM*

Aspect composition (users)

- Two different users may want to add services to the `/etc/services` file without conflicting

```
services.list mADD(foo)  
services.item_foo service A
```

```
services.list mADD(bar)  
services.item_bar service B
```

- These may occur in different (source or header) files, managed by different people
- Although the “result” may be a single configuration file in the target system

Aspect composition (systems)

- We may want to replace a server with a different model, but keep the same services”

```
#include <hardware/modelA.h>  
#include <dnsmaster.h>
```

```
#include <hardware/modelB.h>  
#include <dnsmaster.h>
```

- We could no do this using backup/restore (maybe the machines need different drivers/disks)
- Parts of the server configuration may be conditional on the hardware

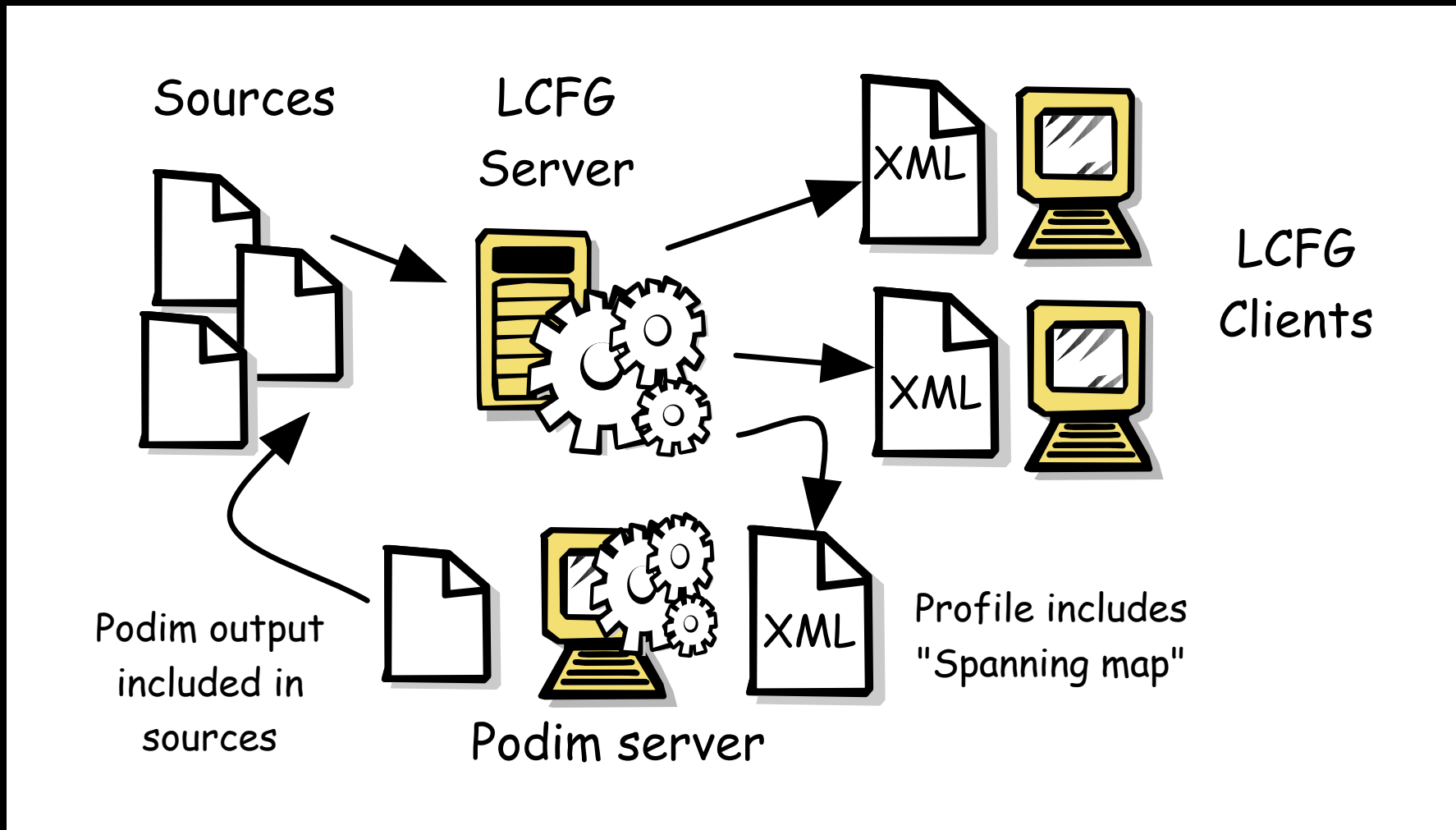
Managing relationships

- A **spanning map** allows the compiler to collect resource values from a set of machines and make them available as part of the configuration of some other machine
 - *the firewall machine knows about the web servers and can create the appropriate holes automatically*
 - *the dhcp server knows the MAC addresses of the clients automatically*
 - *the nagios server knows which services are running on which other machines*
- Resource disappear from the map automatically when the client is removed

Change control

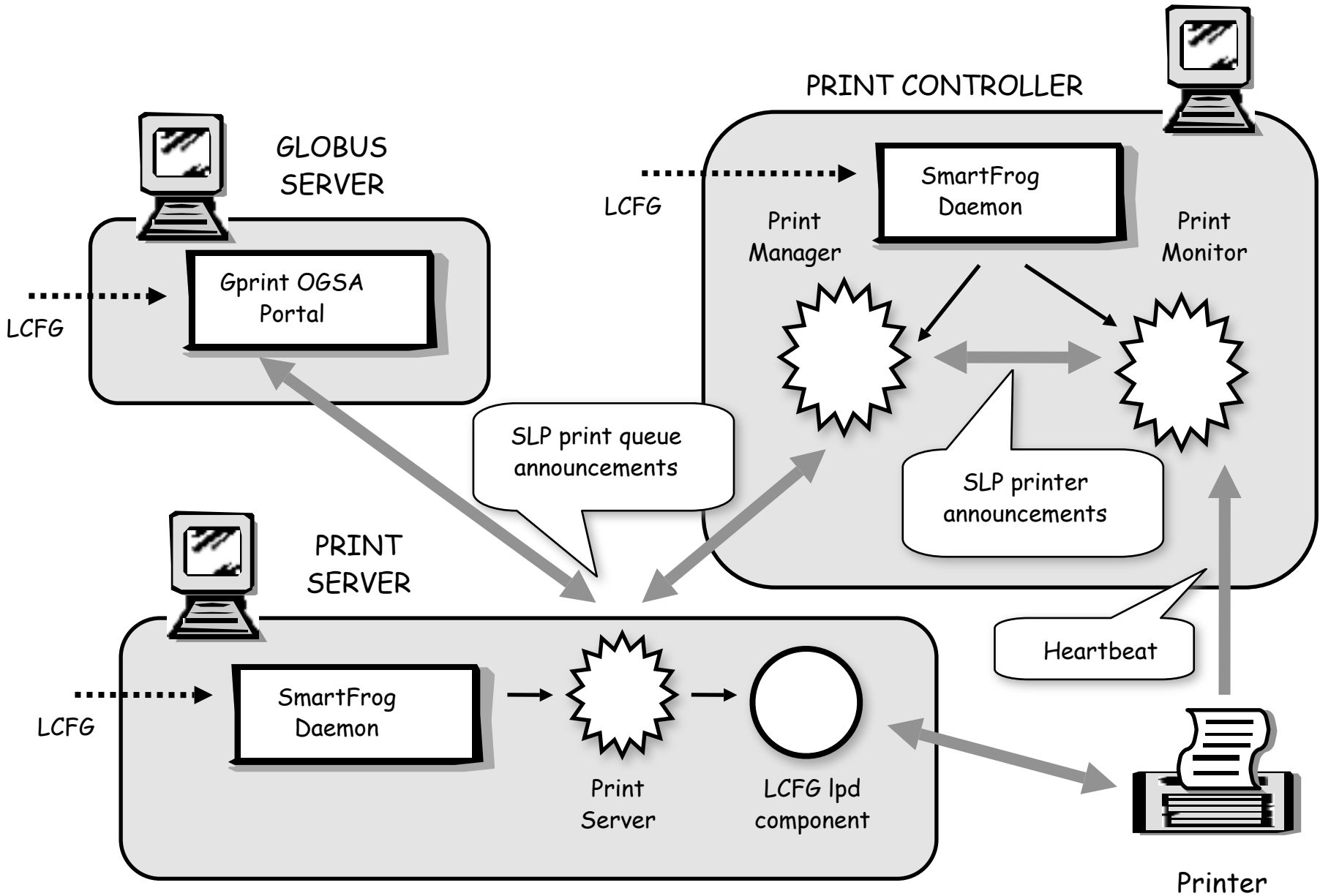
- The source files can be stored in an SVN repository
- The entire site can be reconfigured to its exact state at any time in the past
- We maintain three classes of machine
 - *development (latest configuration)*
 - *test (weekly snapshot of stable developments)*
 - *production (test release validated on set of machines)*
- Some changes need to be “live” and bypass the release system
 - *network changes, server changes*

LCFG/Podim



- A Paper with Thomas Delaet (ICN 2008)
 - Solves “2 DHCP servers on each subnet”

LCFG/SmartFrog






















































































Some open issues

- LCFG doesn't address deployment ordering well
- There is no good model of services
 - *we only deal in “low-level” resources*
 - *this is an advantage and a disadvantage*
- The centralised compilation can be a bottleneck
- The code base is old and difficult to extend
 - *eg. we would like better spanning map support*
- Synchronising changes in resources to updates in the related software is not easy
- There can be a steep learning curve
 - *with cultural problems!*












Examples

- Status pages at <http://lcfg.inf.ed.ac.uk/cgi/>
 - *host list*
 - *component list for one host*
 - *active resource values*
 - *error logs*
- A component skeleton

Status page

    	balhousie	17/10/08 14:23:59	XML
    	ballingry	17/10/08 14:23:42	XML
    	balmalcolm		XML
    	balmullo	17/10/08 14:23:25	XML
    	baltzar	17/10/08 14:23:11	XML
    	bambam	17/10/08 14:25:04	XML
    	banana	17/10/08 14:22:46	XML
    	bankton	17/10/08 14:24:40	XML
    	barker	08/10/08 13:59:33	XML
    	barkly	17/10/08 14:24:27	XML
    	barnyards	11/09/08 09:39:11	XML
    	bauble	17/10/08 14:24:35	XML
    	beath	10/10/08 10:26:10	XML
    	beatles	17/10/08 14:22:45	XML
    	beck	17/10/08 14:24:24	XML
    	beggarsknowe	17/10/08 14:25:24	XML
    	belle	17/10/08 14:25:44	XML

Node page

  	autoreboot	[log] [status] [resources] [doc]
  	boot	[log] [status] [resources] [doc]
  	client	[log] [status] [resources] [doc]
  	condor	[errors] [log] [resources] [doc]
  	cron	[log] [status] [resources] [doc]
  	cyrussasl	[log] [status] [resources] [doc]
  	dhclient	[resources] [doc]
  	dns	[log] [status] [resources] [doc]

Resource values

condor @ camberwell.inf.ed.ac.uk : resources



admin	= iainr@inf.ed.ac.uk
config_condorids	= CONDOR_IDS=2.2
config_executelogin	= EXECUTE_LOGIN_IS_DEDICATED = false
config_filesystemdomain	= FILESYSTEM_DOMAIN=inf.ed.ac.uk
config_flockcollhosts	= FLOCK_COLLECTOR_HOSTS = \$(FLOCK_TO)
config_flockfrom	= FLOCK_FROM=focke.inf.ed.ac.uk
config_flockneghosts	= FLOCK_NEGOTIATOR_HOSTS = \$(FLOCK_TO)
config_flockto	= FLOCK_TO = focke.inf.ed.ac.uk
config_hyperthread	= COUNT_HYPERTHREAD_CPUS = false
config_keyboard	= KeyboardBusy = KeyboardIdle < (15 * \$(MINUTE))
config_readallow	= HOSTALLOW_READ = *.inf.ed.ac.uk
config_scratch	= LOCAL_DIR = /disk/scratch/condor/local
config_uiddomain	= UID_DOMAIN=inf.ed.ac.uk

Errors

condor @ camberwell.inf.ed.ac.uk : errors



```
02/10/08 15:34:05: Could not find Condor executables in directory /usr
```

condor @ camberwell.inf.ed.ac.uk : errors



A component skeleton

```
. /usr/lib/lcfg/components/ngeneric

Configure() {
  echo $LCFG_demo_foobar >my_config_file
}

Start() {
  # Start daemon here
}

Stop() {
  # Stop daemon here
}

# Dispatch methods
Dispatch "$@"
```

Further information

- LCFG - A practical tool for system configuration (the SAGE booklet)

http://www.sage.org/pubs/17_lcfg/

- LCFG web site

<http://www.lcfg.org>

- *includes a VM image which can use to follow the tutorial in the book without installing any native software*
- *includes a mailing list and Wiki*

Paul Anderson
dcspaul@ed.ac.uk