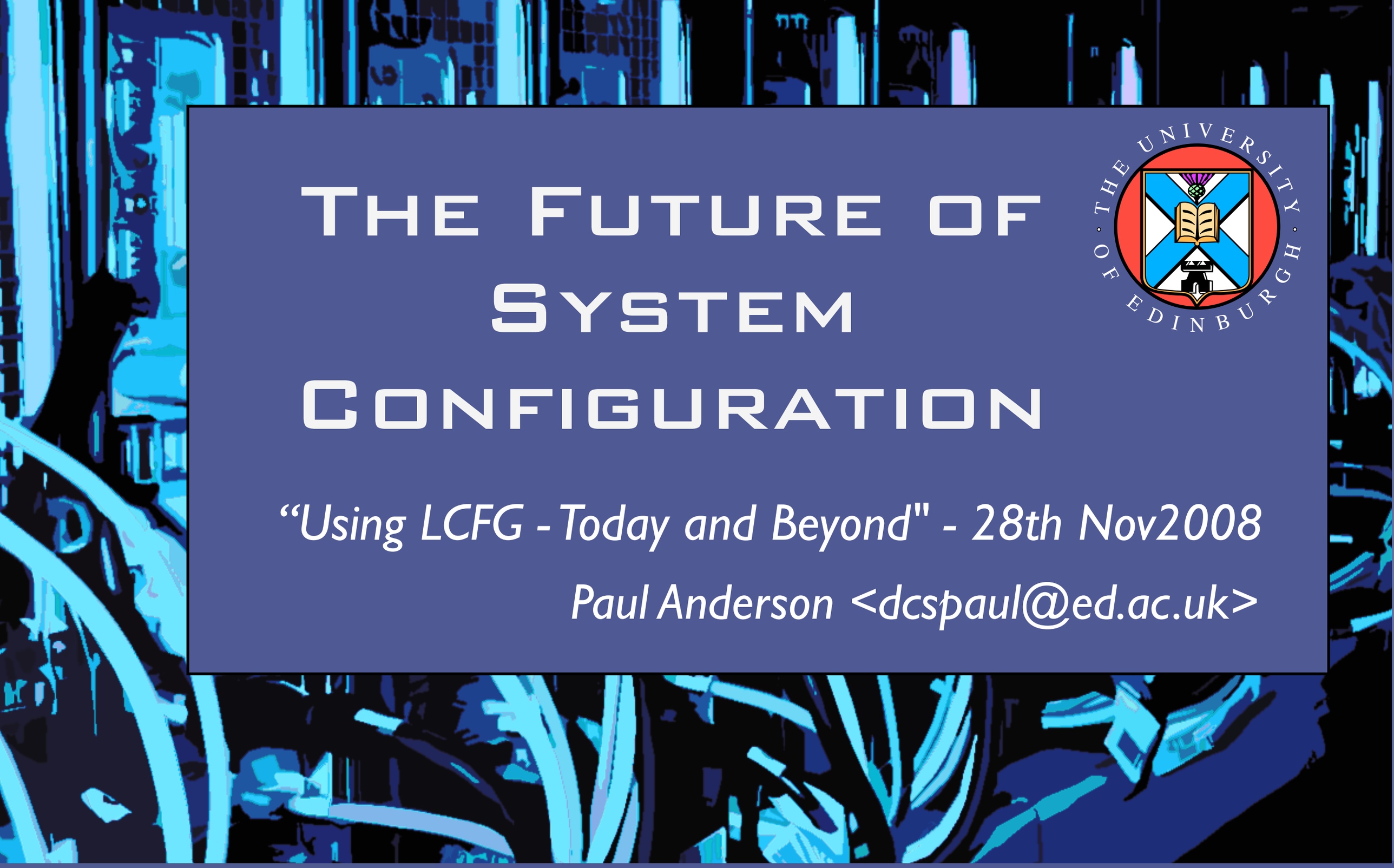


The Future of System Configuration

"Using LCFG - Today and Beyond" - 28th Nov2008

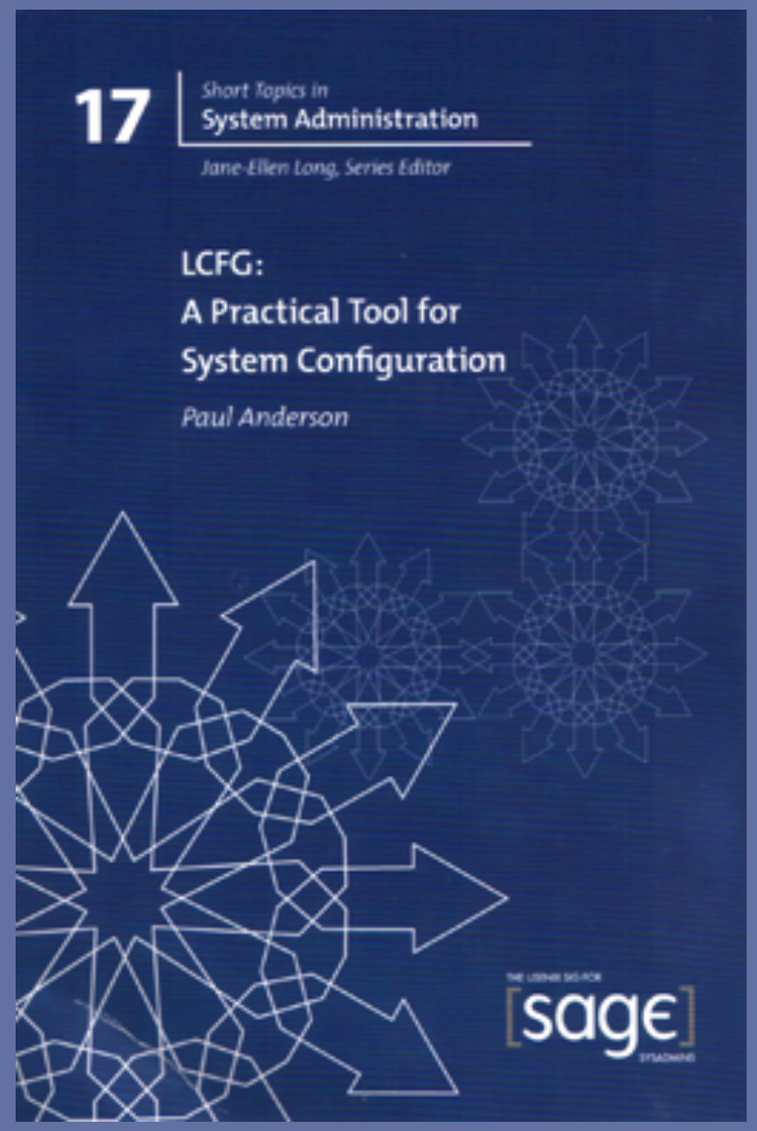
Paul Anderson <dcspaul@ed.ac.uk>

LCFG

http://homepages.inf.ed.ac.uk/dcspaul/publications/lcfg-2008-talk.pdf

LCFG Development

- Original concept dates from 1994
  - *re-implemented about 2001*
  - *stable code*
    - but not easy to work with
  - *a major re-factoring planned*
    - but no new functionality
- Not widely used externally
  - *very little promotion*
    - not a priority
  - *internal use increasing*
  - *steep learning curve*
    - now improved with the book



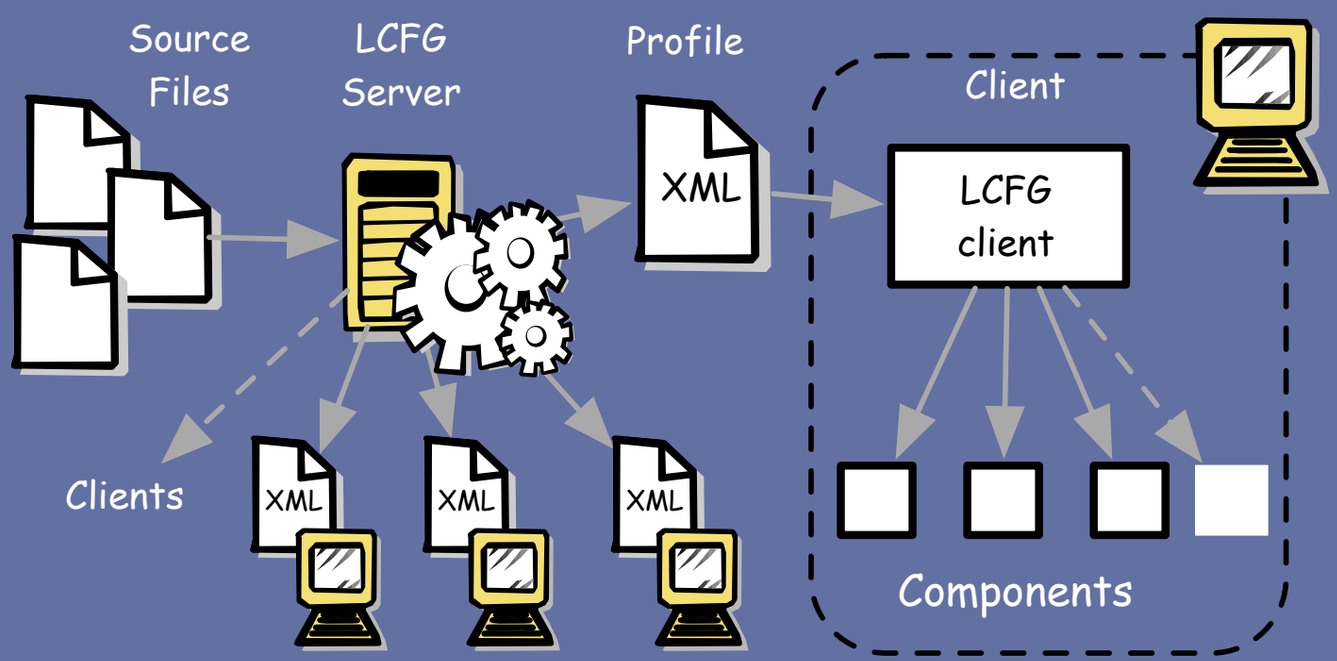
`http://www.sage.org/pubs/17_lcfg/`

Related Developments

- The demand/interest in configuration tools has increased considerably in the last couple of years
- Comparable practical tools have a roughly similar level of functionality to LCFG -
  - *Cfengine (1997), BCFG (2003), Puppet (2005)*
- There is plenty of related work involving more complex/structured frameworks -
  - *CIM, SmartFrog*
- But this has not really affected the configuration tools used by practical system administrators
  - *why not? perhaps these need to be more "agile" ?*

Possible Directions

- Raising the level ?
- Better languages / interfaces ?
- Distributed and devolved management ?
- Autonomics ? Virtualisation ?



RAISING THE LEVEL

(1) Copy this disk image onto these machines

(2) Put these files on these machines

(3) Put this line in sendmail.cf on this machine

(4) Configure machine X as a mail server

(5) Configure machine X as a mail server for this cluster (and the clients to match)

(6) Configure any suitable machine as a mail server for this cluster (and the clients to match)

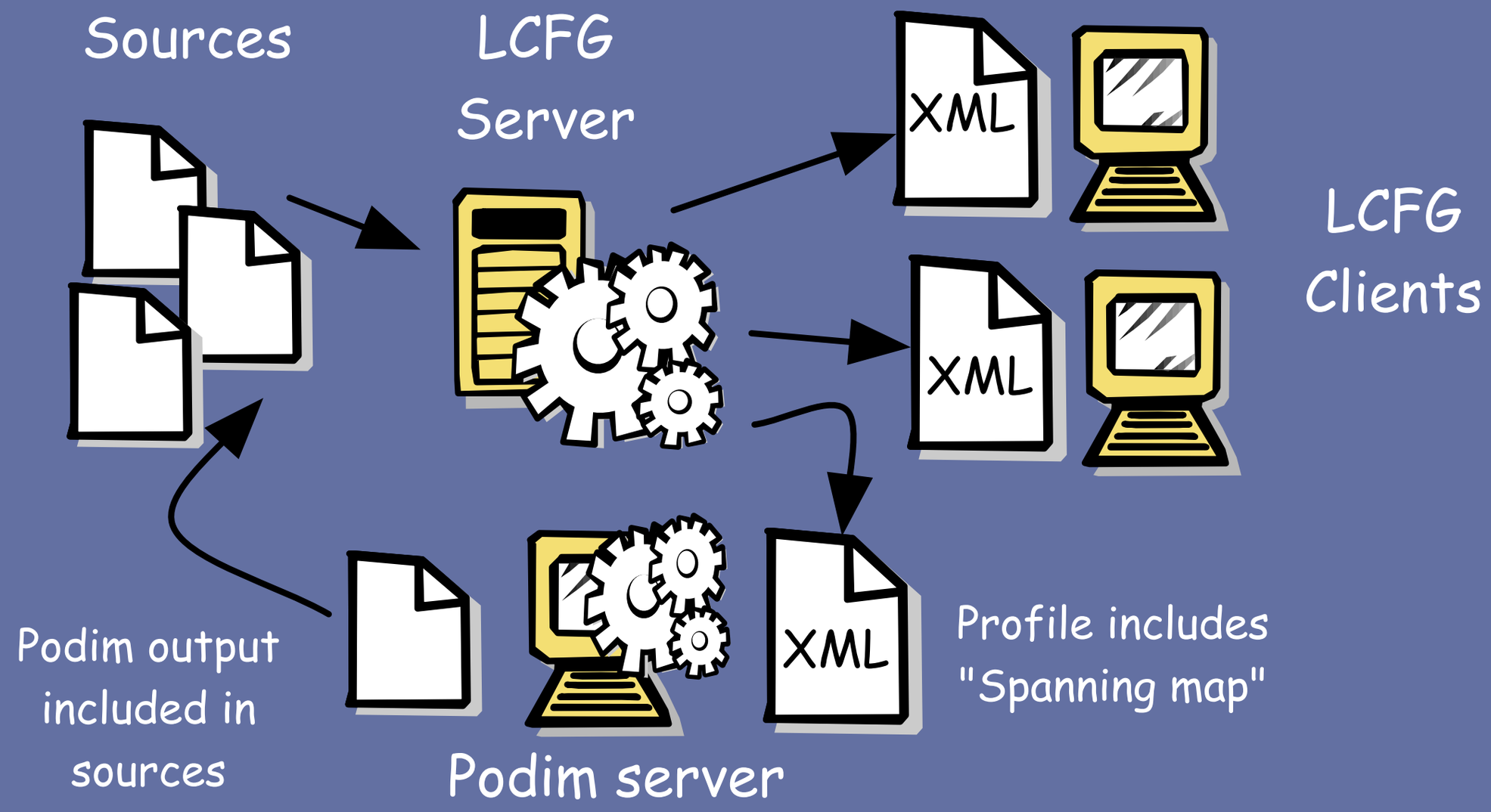
(7) Configure enough mail servers to guarantee an SMTP response time of X seconds

Constraints

- The language forces explicit values to be specified:
  ➡***Aspect A:*** *Use server Y*
  ➡***Aspect B:*** *Use server X*
  This conflict is irreconcilable without human intervention because we don't know the intention

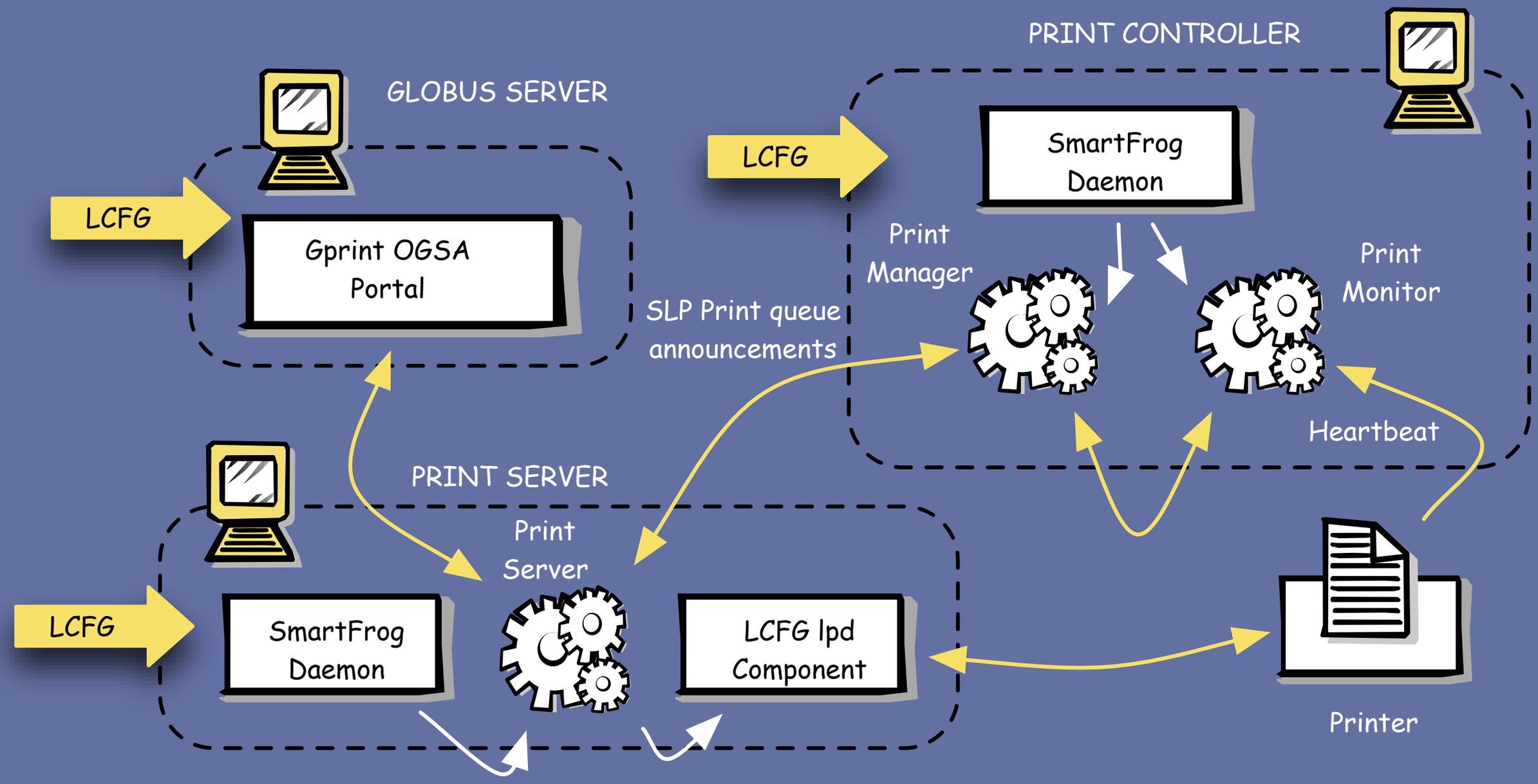
- The user really only wants to say …
  ➡***Aspect A****: Use any server on my Ethernet segment*
  ➡***Aspect B****: Use one of the servers X, Y or Z*
  These constraints can be satisfied by using Y (assuming Y is on the right segment)

LCFG/Podim



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

LCFG/SmartFrog



- A Paper with HP (LISA 2003)

Some Challenges

- Some hard technical problems
  - *as we have seen ....*
- No standards
  - *tools don't inter-operate*
- Evolution is difficult
  - *upgrading a configuration tool is a huge undertaking*
- Trust is important
  - *security, confidence in correctness, and explanation*
- A wide range of people are involved
  - *with different skills and experience*
- Different sites have very different priorities

Autonomy

- The centralised LCFG model is not really appropriate
- Technically ...
  - *for scalability*
  - *and robustness*
- But also because more autonomy is needed ...
  - *for mobile virtual machines*
  - *for laptops and personal machines*
- There is no good solution to this
  - *mobile agents are an interesting research approach*
  - *we need to accept less certainty and "control"*
  - *individual services negociate their configuration*

A Possible Framework

- I've been thinking about a "framework" that would allow different approaches to be mixed ....
- Decisions are made by a combination of human and automatic processes -
  - *the system may present alternatives for the user to select*
  - *decisions may be passed to other (remote) users, or delegated to automatic processes*
  - *"canned" solutions may be stored for configurations or plans*
  - *the user may make explicit choices to constrain automatic solutions*
- Different tools would be possible for deployment

Suggestions?

Paul Anderson <dcspaul@ed.ac.uk>

LCFG

http://homepages.inf.ed.ac.uk/dcspaul/publications/lcfg-2008-talk.pdf