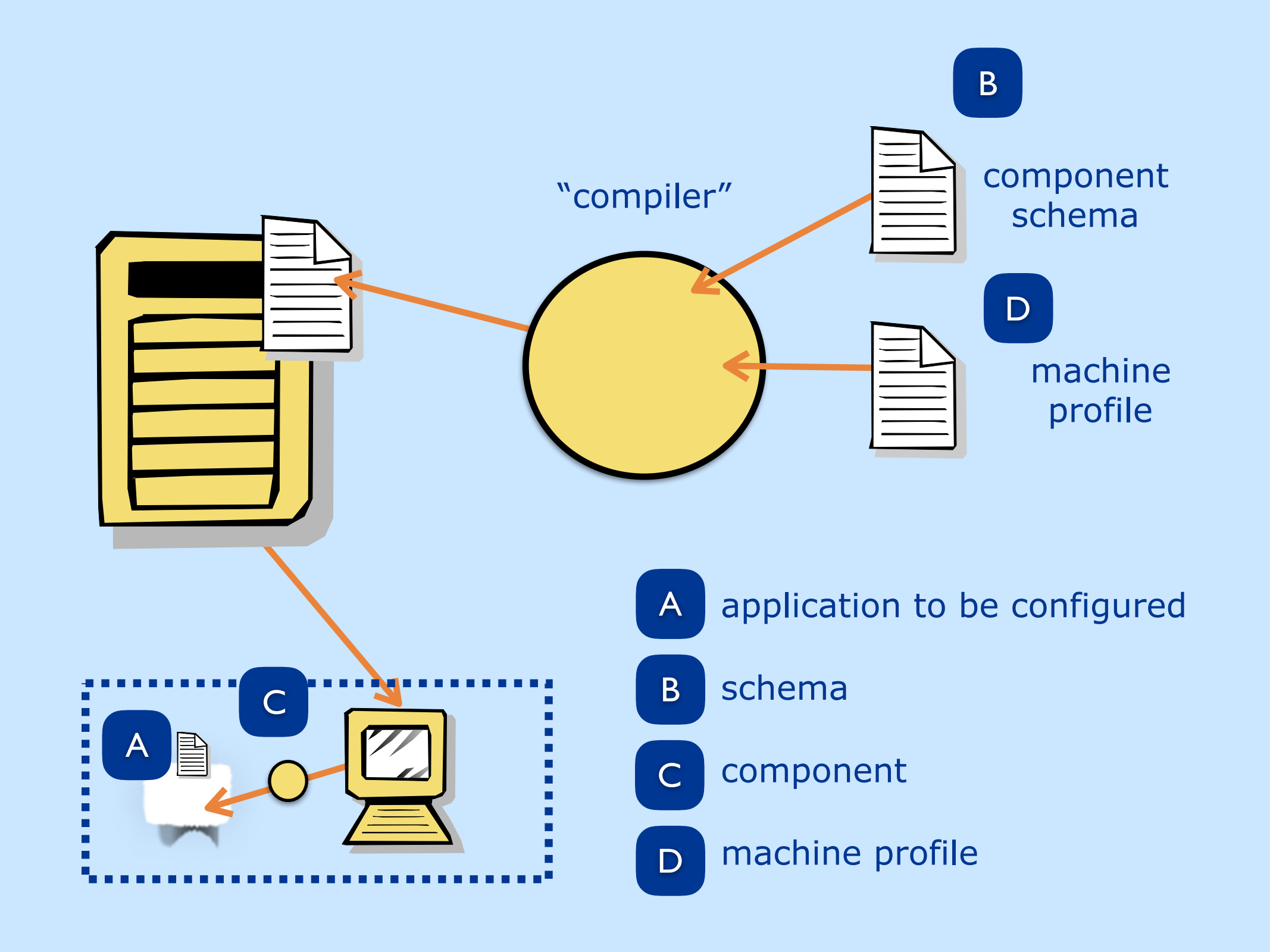
Writing LCFG components

Innovative Learning Week
Wed 17th February 2016

Paul Anderson
Alastair Scobie
Stephen Quinney
Kenny MacDonald

LCFG
Large Scale UNIX Configuration System


B
component schema
"compiler"
D
machine profile
A application to be configured
B schema
C component
D machine profile
C
A

A application to be configured - "chatterd"

/etc/chatterd

write a message to the log every 2 seconds

```
#!/bin/sh

# The message to log comes from the command line argument
message=$1

# Save the PID of the daemon so we can stop it when we want
echo $$ >/tmp/chatterd.pid

# Log the start
echo `date` : chatterd starting >>/var/log/chatterd

# Chatter away
while true ; do
 echo `date` : $message >>/var/log/chatterd
 sleep 2
done
```

nothing LCFG-specific here!
just a simple demo application which is easier to configure than a "real" application such as a web server ...

B schema

/usr/lib/lcfg/defaults/server/chatter-1.def

```
/*
 * LCFG chatter component : default resources
 */

#include "ngeneric-1.def"
#include "om-1.def"

schema 1

message hello world
```

adds some standard resources

the schema version

the default value

the resource name

this tells the compiler what resources are supported

C component - “chatter”

/usr/lib/lcfg/components/chatter

```
#!/bin/sh

# Include the LCFG library functions
. /usr/lib/lcfg/components/ngeneric

# Called when configuration changed
Configure() { … }

# Called to start component
Start() { … }

# Called to stop component
Stop { … }

# Main function dispatches the methods
Dispatch "$@"
```

C component - "chatter" - Start()

```
# Called to start component
Start() {

  # only start it if it isn't already running
  # i.e. if the PID file does not exist
  if ! test -f /tmp/chatterd.pid ; then

    # start the application as a background process
    /etc/chatterd "$LCFG_chatter_message" &

  fi
}
```

be careful not to start multiple processes!

the framework makes the resources available as variables with names: $LCFG_component_resource

pass the value of the message resource as an argument to the process

the "&" starts the chatterd application as an asynchronous background process

component - “chatter” - Stop()

```
# Called to stop component
Stop() {

  # if the PID file exists, it is running ...
  if test -f /tmp/chatterd.pid ; then

    # read the process ID from the file
    PID=`cat /tmp/chatterd.pid`

    # kill the process
    kill $PID

    # remove the PID file
    rm /tmp/chatterd.pid

  fi
}
```

C component - “chatter” - Configure()

```
# Called when configuration changed
Configure() {

  # if the PID file exists, it is running ...
  if test -f /tmp/chatterd.pid ; then
    Stop
    Start
  fi
}
```

if the process isn’t running, there is nothing to do ...

the Start() method will take care of restarting chatterd with the new value of the message resource

D profile

/var/lcfg/conf/server/source/localhost

```
#include <local/site.h>
#include <lcfg/os/minimal_sl6.h>
#include <lcfg/hw/virtualbox.h>
#include <lcfg/options/lcfg-server.h>

!profile.components         mADD(chatter)
profile.version_chatter     1

chatter.message              I'm Working!
```

add the component to the profile

explicit version numbers support upgrading with schema changes

the default from the schema will be used if you omit this

Resources

You will need to install the following ...

- /etc/chatterd
- /usr/lib/lcfg/components/chatter
- /usr/lib/lcfg/defaults/server/chatter-1.def

Copies are available from here:

- http://www.lcfg.org/ilw2016/

The first two will need to be executable:

- chmod oug+x /etc/chatterd
- chmod oug+x /usr/lib/lcfg/components/chatter

You will also need to edit ...

- /var/lcfg/conf/server/source/localhost

Try ...

Check the resource values

- qxprof chatter

Start & stop the component

- om chatter start
- om chatter stop

Check the output

- cat /var/log/chatterd

Reconfigure

- om chatter start
- check the output
- change the message
- the application should reconfigure and start logging the new message

Further suggestions ...

Implement some more resources

- the logging frequency
- the name of the message file

Browse some real components

- /var/lib/lcfg/components/ntp
- /var/lib/lcfg/components/mail

Subsystems with an imperative interface

- Some subsystems have to be configured using imperative commands, rather than a “declarative” config file
- Eg. rpms, or the Ubuntu firewall (ufw)
- Think about how the component can be used to present a declarative interface to the configuration system

Further Topics

Topics not covered (see the book) ...

- “tag lists” and list order
- “spanning maps”
- managing lists of installed packages
- “prescriptive” vs “lightweight” configuration
- installing machines from scratch
- managing an entire site

http://www.lcfg.org/ilw2016/