Towards Object Oriented System Administration

Ronan.Keryell@enst-bretagne.fr

Attila SAORIN FERENCZ

—

Laboratoire Informatique & Télécommunications

Département Informatique

École Nationale Supérieure des Télécommunications de Bretagne

Summary 1

Work in progress

We present some thoughts to introduce an object oriented approach in large system administration in order to ease automatic deployment, to structure the organization and to able open system infrastructure databases for people that needs to build up new system infrastructures from scratch or to collaboratively administrate some networks.

The infrastructure itself is based on a policy, a set of reference files and an object oriented and extensible tool PCFengine that extends the concepts found in Cfengine tool (http://www.cfengine.org) that can be seen like an autonomous agent acting as a computer immune system that keeps the computer running.

PCFengine can be seen as many instances of Cfengine directly implemented in an object oriented language (Python). Whereas in a Cfengine a specific language is used to describe the actions to be

executed according a predicate or a context, in PCFengine all is reified and is a direct executable Python code. In this way PCFengine benefits from the powerful object-oriented and scripting language Python and can be easily configured or extended with multiple inheritance, evaluation, introspection and functional flavor.

Introduction 3

- Computer science is 60 year old
- Computers are everywhere
- ... But mostly administrated without computer science concepts !
  - ► Manual installation
  - ► Hardware and software babysitting
- Do not scale very well : what for large communication and/or computer networks, for grid infrastructures, active networks ?
- ⇝ Bootstrap computer administration itself

Apply computer science methods to computer administration

- Everybody is faced to infrastructure deployment
- ... and often to a non-infrastructure deployment ☹
- Some labs throw people in manual installation of their own computer ☹
- Most of the cost of desktop ownership is labor
- Need to automate the process
- Build some open example infrastructures
  - ► Freely available
  - ► Collaborative internal (and external) administration
  - ► Capitalize distributed knowledge
  - ► Open to external contributors
  - ► But need a censoring system for confidential information ☹ :

passwords, cryptographic keys, some personal information,...

⇝ OASIS Project ?

- Policy
  - Describes the system organization
  - How does it work ?
  - What is dealt or not ?
  - Who does what ?
- Tool(s) and configuration files
  - Do the real work
  - The tools use some configuration files (≈ the program) but also some reference files (≈ the data)
- Reference files
  - OS-specific files
  - Hardware-specific files

- Site-specific files
- Subnetwork-specific files
- Project-specific files
- Personal files
- Invariant files
- ...

Automatic installation 8

- Each (serious) operating system has an automatic installation procedure
  - ► Windows (unattended install)
  - ► Solaris (JumpStart)
  - ► Linux/RedHat (KickStart)
  - ► Linux/Debian (FAI)
  - ► *BSD
- All have their own configuration and reference files
- $\rightsquigarrow$ Always need some tools to polish the corner
- A tool that generates automatic installation configurations (in a portable way...) may help...

ISCONF 9

`http://www.infrastructures.org/`

- Scalable, flexible, and rapid deployments and changes
- Cost effective, timely return on IT investment
- Low labor headcount
- Secure, trustworthy computing environments
- Reliable enterprise infrastructures

Tools : CVS, SUP, `make` to keep state (a machine is a `make` target)

ISCONF advantages 10

- Remote management
- Secure communications
- Distributed Monitoring
- Unattended network-based installation
- Automatic host administration (no need to manually track or apply changes to managed hosts)
- Unified desktop and server management
- Single System Image
- Single Signon
- Continuous, long-term live host management (no re-installation needed to apply upgrades)

ISCONF advantages 11

- Ordered, validated changes to any given host
- Prototype and class-based host definitions
- A coherent framework for managing all of the above
- What if there is shift between `make` state and reality ? ☹

Arusha project 12

`http://ark.sf.net/`

- Tools
- $\exists$ real policy used and given as example (Sidai)
- Try to be a sysadmin equivalent to open-source software development
- Sysadmin is about making effective users, not just effective systems
- All things are objects with attributes and methods : machines, users, 3th-year students, maintenance contracts, mailing lists, network ingress points, user environments,...
- It's the *source code* of a site
- XML object configuration

Arusha project 13

- ARK macro-language
- Quite interesting philosophical point of view

`http://www.cfengine.org`

- Computer immune system that keeps the computer running
- Cfengine follows a class-based decision structure to trigger actions
- Some basic actions
  - ► Check and configure the network interface
  - ► Edit text files
  - ► Make and maintain symbolic links, including multiple links from a single command
  - ► Check and set the permissions and ownership of files
  - ► Tidy (delete) junk files which clutter the system
  - ► Systematic, automated mounting of filesystems (Unix)
  - ► Checking for the presence of important files and file systems

- ► Controlled execution of user scripts and shell commands

File `sendmail.cf`

```
groups:
   mail_servers = ( gavotte.enstb.org smtp-cri.ensmp.fr minou.lit.enstb.org plinn.lit.enstb.org )

copy:
   solaris::
      /usr/local/sbin/sendmail
         dest=/usr/lib/sendmail
         type=byte
         define=RelaunchSendmail
[...]
      $(shared_conf)/mail/$(site)/access.pag
         dest=/etc/mail/access.pag
         owner=root group=root
         type=byte

   mail_servers.solaris::
      $(shared_conf)/mail/$(site)/local-host-names
         dest=/etc/mail/local-host-names
         type=byte
         define=RelaunchSendmail
[...]
```



```
   debian.!mail_servers::
      $(shared_conf)/$(site)/$(os)/etc/mail/leaf-sendmail.mc
         dest=/etc/mail/sendmail.mc
         owner=root group=root
         type=byte
         define=RelaunchSendmail
[...]
editfiles:
   debian::
      { /etc/mail/submit.mc
         AppendIfNoSuchLine "include(‘/etc/mail/starttls.m4’)dnl"
         DefineClasses "RelaunchSendmail"
      }

files:
   solaris::
      # The statistics file for sendmail:
      /etc/mail/statistics
         mode=644 owner=root group=bin
         action=touch

links:
   solaris::
```


```
      # Links for sendmail pseudo-commands:
      /usr/bin/hoststat ->! /usr/lib/sendmail
      /usr/bin/mailq ->! /usr/lib/sendmail
      /usr/bin/newaliases ->! /usr/lib/sendmail
      /usr/bin/purgestat ->! /usr/lib/sendmail

directories:
   # Set sendmail mode and owner properly for security:
   solaris::
      / owner=root mode=go-w
      /etc owner=root mode=go-w
      /etc/mail owner=root mode=go-w
      /usr owner=root mode=go-w
      /var owner=root mode=go-w
      /var/spool owner=root mode=go-w
      /var/spool/mqueue owner=root mode=go-w
      /var/spool/clientmqueue owner=smmsp group=smmsp mode=770

shellcommands:
   RelaunchSendmail.solaris::
      "/etc/init.d/sendmail stop"
      "/etc/init.d/sendmail start"
```



```
   RelaunchSendmail.debian::
      # There is a bug in sendmail 8.12.3-4
      "/bin/sh -c 'yes | (cd /usr/share/sendmail; sendmailconfig)'"
      #"/etc/init.d/sendmail reload"

processes:
   any::
      # Start it anyway. More than 0 instance should run:
      "sendmail"
         matches=>0
         restart "/etc/init.d/sendmail start"
```


Conclusion on Cfengine 20

- Good concepts : autonomous immune agent
- Hard to extend
- No modularity
- No programming power
- Yet another language to learn...
- Convergence can not be proved... Life neither ! ☺

Object oriented methods 21

- Privilege design and concepts instead of raw performance (no need to PFLOPS here !)
- Machines and routers can be classified according various categories
  - ► Unix or Windows or...
  - ► Core or edge router
  - ► Server or end-user machine
  - ► Computer of a team A or B, of a site X or Y
  - ► WWW server or mail server
  - ► ...
- Use inheritance to specialize machines or roles
- Use multiple inheritance as machines can have various roles

- Inheritance can be used to specialize machine « classes » *and* to specialize actions in a local context

Expressiveness 23

- Reference files are (inert) data
- Configuration files lack programmability
- Need modern computing expressiveness
- Clear infrastructure $\rightsquigarrow$ high modularity

Development language 24

- Used as the configuration language itself seen by the user $\rightsquigarrow$ not the *yet another weird language* syndrome
- Object oriented language with multiple inheritance : extensibility and natural classification
- Scripting power for system administration and easy gluing of various system components
- Well-supported on various architecture to serve as the portability tier
- Functional flavor, evaluation and introspection

Python

Less well-known : OCAML

PCFengine 25

- Python code
- Just import the PCFengine library
- Use Cfengine concepts (but use *predicates* instead of *classes*)
- Full-fledged programming language

Actions predicated by context or conditions

- `Predicate("linux and (LIT | ReActiVE)")` (Cfengine style)
- `Predicate(And(Predicate("linux"),Predicate("Hour_3")))` (algebric style)
- `Predicate(lambda () : compute-the-captain-age)` (functional style)
- Static or dynamic predicate
  Does the time (or system) evolves ?
  $\rightsquigarrow$ add `dynamic = true`


```
#!/usr/bin/python2.2

from engine import *

pcfengine = PCFengine()

pcfengine.def_Predicate('LIT')
my_machines = Predicate('sun4 or (linux and not bsd)')
web_server = Predicate('www')

pcfengine.set_Current_Predicate('linux')

pcfengine.add_Action(Copy(src = '/home/attila/python/texto.txt',
                          dst = '/home/attila/toto.txt'))
pcfengine.add_Action(Edit(file_name = '/home/attila/toto.txt',
                          action = [ Edit.Replace('metodo comun.',
                                                  'VAMONOS DE JUERGA'),
                                     Edit.Call(my_edit)
                                   ]))
def my_edit(context) :
    if context.Match('A[b|c*]') :
    context.Append('#I have found ' . context.getMatch())
```


PCFengine example 28

```
    else :
        warning('This file is clearly corrupted :' . context.getBuffer())
    return true

pcfengine.add_Action(CreateDir(path = '/home/attila/chorraditas/'),
                     predicate = my_machines)
pcfengine.add_Action(Process('apache', owner = 1, chdir = 1)
                     predicate = web_server)
pcfengine.add_Action(Edit(file_name = '/etc/shadow',
                          action = Edit.Call(set_WWW_passwd)))

addDistributedFaultTolerantUserFileServer('gavotte')

addMachineInInfrastructure('rodomouls', system = 'Solaris')
addMachineInInfrastructure('plinn', system = 'Debian')

print pcfengine.execution()
```


Extension 29

- Trivial to add new actions, new edit actions
- Deriving new actions

```
class EditWithRCS(Edit):
      def saveFile() :
          Check out the file
          Edit.saveFile() :
          Check in the file
```

  ⇝ Document many internal *hooks*
  Aspect programming may help

- Adding machines
  - ► A machine is not a class but a predicate
  - ► Scatter this predicate in other predicates (`www_server`,...)
  - ► OK since predicates are mutable objects
- Unifying inheritance mechanism for actions and machines ?

`pcfengine.execution()` run a PCFengine cycle and return *true* if something has changed... and may need some other work

Fix point:

```
while pcfengine.execution() :
      pass
```

Hmm... Well, non computable... ☹

```
N = 0
while pcfengine.execution() :
      N = N + 1
      if N > 1000 :
         break
```


Reference files : how to map instance classes on file systems ?31

- Configuration ≡
  - ► Description and script languages
  - ► Configuration files
- How to map configuration files associated to classes and class instances on a file system hierarchy ?
- How to map configuration files in a sensible way for ?
- If the description and script languages are also seen as configuration languages : same problem as above

- Different views in `/usr/local/share/conf/ref`
  - Application-centric

```
OpenOffice/
  etc/
  usr/
  var/
sendmail
  etc/
  var/
```

    Interesting if we want to install a new application quickly
  - Site-centric

```
ENSTBr/
  etc/
  usr/local/share/emacs/site-lisp/
ENSMP/
  etc/
  usr/local/share/emacs/site-lisp/
```

    Interesting if we want to install a new site quickly. Need also

    more refinement for labs, networks,...
  - OS-centric

```
Debian/
  etc/
  var/
Mandrake/
  etc/
  var/
Solaris/
  etc/
  usr/
  var/
```

    Interesting if we want to install a new OS quickly

- In the real life it's a mix of all these
  $\rightsquigarrow$ How to hierarchize these views ?
- Should it commute ? Using a database view or semantic file system ?

Different views of reference files 34

- Hmmm... Trying to keep Unix philosophy : everything is a file

Conclusion 35

- Work in progress... (for 15 years ☹)
- Phosphorer avec les gourous « objets » et système locaux
- Forger le chaînon manquant entre un système réparti objet et son déploiement physique
- Bah, tout ça n'est qu'un médium après tout ☺
- Vers une action incitative GET ?

Table des transparents