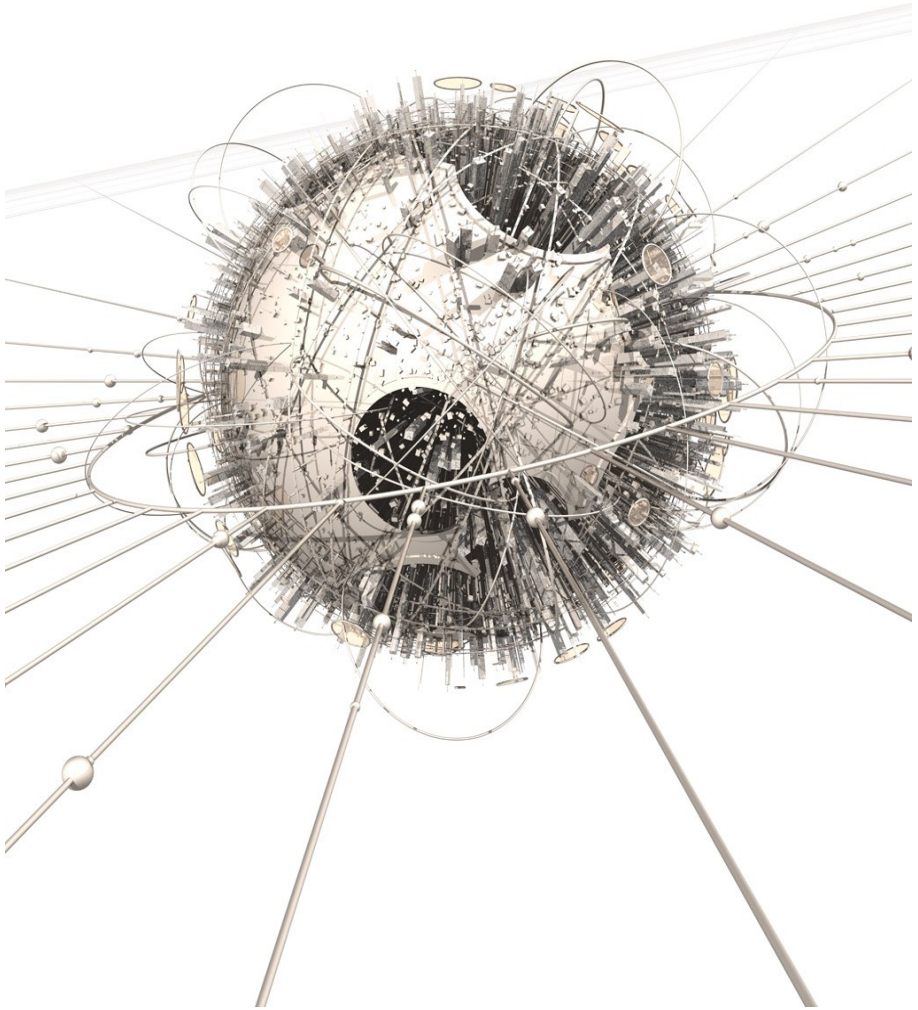
The Future of Free/Open Source Configuration Management

Mark Burgess

Cfengine
&
Dept of Computer Science
Oslo University College

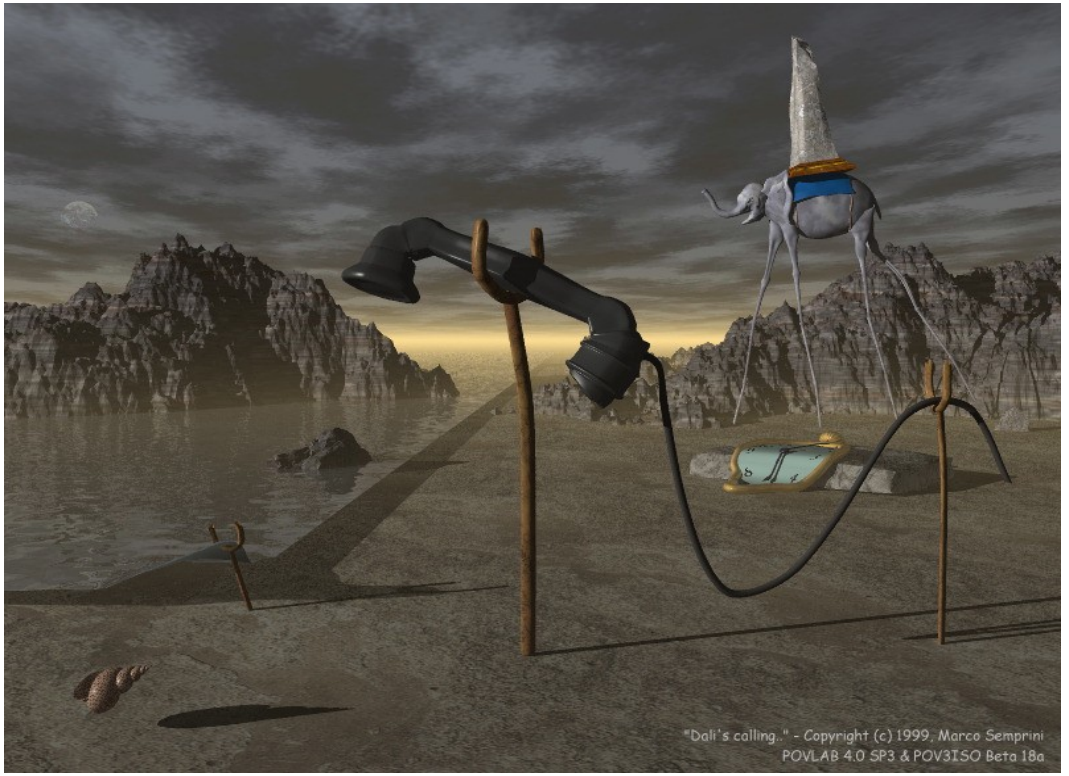
The freedom trend: F = me³



Personal, untethered...

Unseen infrastructure: networks

- Roads
- Tracks
- Electricity
- Cable
- Wireless
- …



- Freedom comes with hidden costs: **CONFIGURATION of the magical infrastructure and mobile freedom**

Freedom principle in action

- Applications drive sysadmin today
- This is just part of the trend
  - Mainframe → workstation → laptop → mobile
  - Integrated systems → shell commands
  - Software-suite → FOSS → “app”
  - Pre-programmed menus → scripts → standalone promises
  - Centralized control → local or federated control

Configuring for “freedom”

- The problems we face are increasing *scale* and *complexity* as freedom is thrust upon us by social or environmental forces – *our desire for flexibility*
- What allows us to handle this complexity?
  - Atomize – keep it simple and light
  - Untether – give me the freedom to work/live
- This is not the way it used to be done in IT, so we need to go back and understand why FOSS config systems are different.

Chapter 1: primæval soup

Going back to the beginning of things

Homework 1:

How do you configure a glass of water?

MODELLING HINT (melt first):

```
molecules:

  "water"
    atoms => { "hydrogen", "hydrogen", "oxygen" };

bonds:

  "hydrogen"
    valency => "+1", # oxidation number
  container => glass;

  "oxygen"
    valency => "-2",
  container => glass;
```

ISOMORPHIC SCENARIO:

```
molecules:

   "computer"
     atoms => { "motherboard", "disk", "disk" };

bonds:

   "motherboard"
        disk_valency => "-2",
     network_valency => "-1",
           container => host_1;

   "disk"
        disk_valency => "+1",
           container => host_1;
```

Homework 2:

Explain the difference between ice and a cloud?

The “promise” model (cfengine)

- Atomic elements + convergence principles are a sufficient description of the problem.
- The principles for managing diversity are:
  - Non-conflicting building blocks
    - (primitive elements) files, processes, database tables, etc
  - Desired maintainable properties (repairable)
    - What basic properties can be promised
  - Stable arrangements (configuration)
    - Converge spontaneously by attraction to desired state

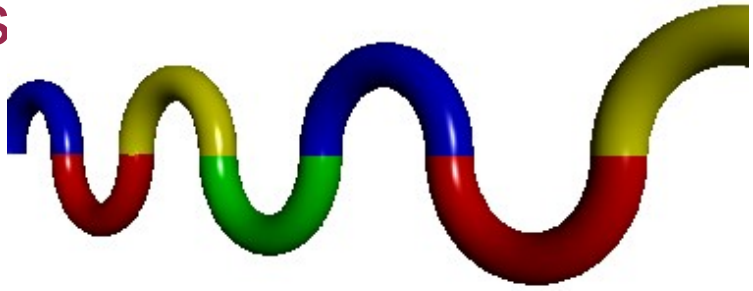
Chapter 2:

Stable recreatable patterns

Exploiting packages and services

High level languages

- After water, interesting complexity emerged...
- Re-digitizing configuration descriptions at a higher level
  - In the history of the world, domain specific languages (DNA/RNA)
- In chemistry, there are multiple languages:
  - Genes (to configure peptides/proteins)
  - Proteins (to configure tissues`



“High level” configuration language

```
peptides:

  "amylin"

      comment => "pancreatic beta cells",
   amino_code =>
              "KCNTATCATQRLANFLVHSSNNFGAILSSTNVGSNTY";

amino_acids:

  "K"
      comment => "Lysine",
       codons => { "AAA", "AAG" };
```

Does this digitization go too far?

- Pre-fab components reduce flexibility!
- Given a supply of amylin, as a black box
  - Can't reconfigure it into Tryptone even though there is an underlying freedom to do so
- Either need an easy supply of all black boxes (off the shelf warehouse) or we've failed the flexibility test
- We constantly trade detail for flexibility
- There is a risk of oversimplifying

Homework 3:

How do you configure a dinosaur?

(bring it up to the present day) Homework 3:

How do you configure AIX?

Or a moose?

Distributed build + repair

stability enables freedom

- Underlying these languages are hidden stable processes
  - Electrical attraction in water
  - RNA/DNA copying fidelity in proteins
  - Jigsaw shapes that fit/don't fit
- The coding of these mechanisms provides local autonomy of build + repair
  - We can distribute the processes in PARALLEL
  - No umbilical required (dissociation)
  - Infrastructure or free-floating mobile device

The Myth of Centralized Management

- Single point of control, single point of failure
  - Serialization and Amdahl's law
- How does centralization actually help?
- Hearts and minds – unviable dinosaurs + old push based technologies

Chapter 3:

The IT-atious era and beyond...

IT automation today (cfengine '93)

- Old push based technologies are out
  - Don't scale
  - Don't support inevitable desire for freedom
- Distributed decentralized network thinking is in
- **Atoms:** files, processes, packages, machines, networks …phone settings (digital “DSL”)
- **Bonds:** stable documented relationships (promises)
- **Self-healing = trusted dependability**

The role of FOSS

- Starting without commercial interest gives freedom to think clearly without pressure
  - e.g. Big four reactive alarm system trap
- Connects ideas to a critical audience to prove new technology
  - Zero price = zero barrier to adoption
  - Source code less important than flexibility
  - Community of writers less important than discussion and verification
- A re-branding of the scientific discourse

Our world: freedom vapours condense around us … the real cloud

- We now know how to to support IT freedoms
  - Invisible infrastructure √
  - Lightweight devices that enable creativity √
  - Servers, phones, pads, apps... (the real cloud)
- This is complexity management, a form of knowledge or information management

Homework 4:
How do you configure a web page?

Chapter 4:

At the KT boundary

Knowledge-Technology
Knowledge-Transfer

The new role of commerce

- Division of labour:
  - Machines: implementation
  - Humans: knowledge
- Agile and re-usable
- No expertise required
- Make black boxes
  - BUT: if you can touch it, it needs to be managed!
- Charge for the value of the simplicity
  - Oversimplifying just holds you back

Freedom + commerce

- Make many small-smart boxes
  - The kinds of promises they need to keep will evolve relevant to the environment of the day...
- Easy to use, simple but powerful
- Seeking appropriate compromises
  - *This is the start of knowledge management*

Redesigning the configuration language for Cfengine 3: free interface design, to conceal without removing configuration complexity: build your own coloured box

Mutating FOSS ideas and habits

- Build and monitoring → integrated services
- Independent monitoring is better?
  - You don't fly a second plane to measure altitude
  - Most monitoring tools do not offer any kind of scientific rigour in measurement anyway
- Doing and Knowing need to come together in a much less ad hoc way
  - Here FOSS loyalties can get in the way of progress

Automation rehumanizes system administration

- ***"Dehumanization is not replacing humans by machines but in making humans act like machines in the first place."***
- The future challenge is now Knowledge
  - Planning, knowing, insight – freedom to change
  - Disseminating, training – freedom to change jobs
  - Quality assuring – freedom to trust not micromanage
  - Understanding the beast you made!
- Good models bring simplicity and agility
- Pedagogical and didactic skills return
  - Humanities students can play an increasing role

Successes and Failures

- Unix commands – freedom to develop
- Apps
- Mammals
- Freedom to change
- Small reliable building blocks

- Dinosaurs – inflexible brute force fragile to catastrophe
- Forced / pushed compliance
- Big black boxes

Final Homework:

How do you configure success?

Summary

Challenge: *Free adaptability, stability and scalable complexity*

Untether dependencies + simple clear promises

→ clear expectations
→ collaboration
→ social ecosystem
→ personal enablement
→ Freedom

The real cloud is all of us

Free Speech or Free Beer?