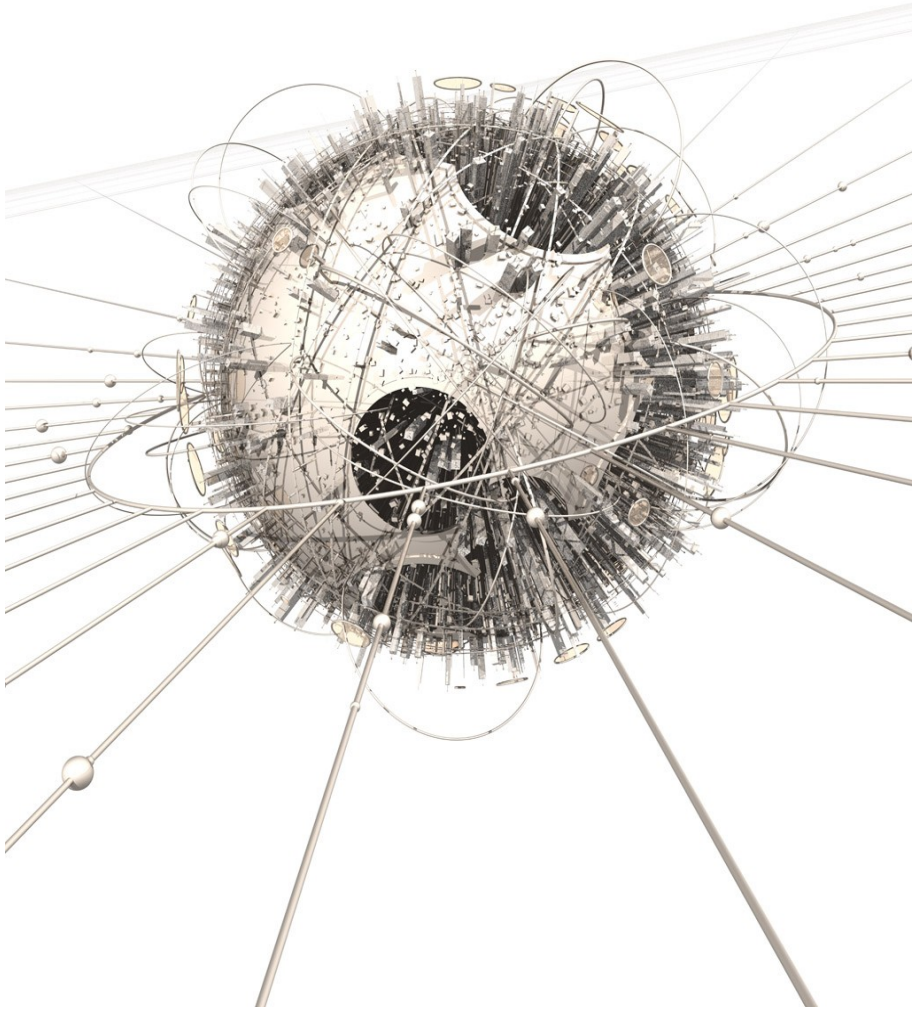
The Future of Free/Open Source Configuration Management

Mark Burgess

Cfengine
&
Dept of Computer Science
Oslo University College

The freedom trend: F = me³



The Freedom principle

- Atomize and untether!
- Mainframe → workstation → laptop → mobile
- Integrated systems → shell commands
- Software-suite → FOSS → “app”
- Pre-programmed menus → scripts → standalone promises
- Centralized control → local control
  - Too much freedom: no control

The commerce principle

- Keep it simple and agile and re-usable
- Broad appeal, no expertise required
- Integrate complex parts into a black box
- Charge for the value of the simplicity
  - Oversimplifying just holds you back

Freedom + commerce

- Make many small-smart boxes
- 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

Configuring “freedom”

- The problems we face are *scale* and *complexity* as freedom is thrust upon us by social or environmental forces
- Infrastructure versus me
  - Atomize
  - Untether
- **Configuration = choice + arrangement of parts**
- Configuration under these circumstances has been done before...let's go back in time

Chapter 1: primæval soup

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?

Cfengine's “promise” model

- The principles for managing diversity are:
  - Non-conflicting building blocks
    - (primitive elements)
  - Connection interfaces
    - (possible bonds)
  - Stable arrangements
    - (configurations)

Chapter 2:

Genes and molecular complexity

Molecular computing

- Re-digitizing configuration descriptions at a higher level
  - Domain specific language (RNA) – speeds up evolution!
- In chemistry, there are two languages:
  - Genes (to configure proteins)
  - Proteins (to configure tissues)

“High level” configuration languages

```
peptides:

  "amylin"

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

amino_acids:

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

Have we gone too far?

- 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 or we've failed
- We constantly trade detail for flexibility
- There is a risk of oversimplifying

Homework 3:

How do you configure a dinosaur?

(translation) Homework 3:

How do you configure AIX?

Self-healing enables freedom

- Underlying these languages are hidden stable processes
  - Atomic electrical attraction
  - DNA copying mechanisms
  - Jigsaw shapes that fit/don't fit
- These mechanisms are stable (self-healing) and provide local independence
  - No umbilical required
  - Can happen *in situ* anywhere conditions are ok

The Myth of Centralization

- Single point of control, but single point of failure
- Hearts and minds?
- How does centralization actually help?

Chapter 3:

The IT-atious era and beyond...

IT management

- **Atoms:** files, processes, packages, machines, networks …
- **Bonds:** stable documented relationships (promises)
- **Self-healing:** don't just try to build, but maintain
- Scripts usually break all these principles
- *Dehumanization is not replacing humans by machines but in making humans act like machines in the first place.*

The role of FOSS

- Not specifically important – it could have happened differently
- Gave people like me a willing audience to introduce 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

Personal freedom condenses around us … the real cloud

- Like it or not we have 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 management

Chapter 4:

At the KT boundary

Knowledge-Technology
Knowledge-Transfer

Environmental diversity

- Diversified infrastructure, supporting freedom
- Hosts not all alike (diversity)
- Break away from totalitarian centralization
  - Companies increasingly see this cannot work
  - Federated approaches
  - Distributed and mobile
  - Clouds and disposable VM resources
- Need powerful modelling to support this
  - This is where Cfengine has remained strong

Refactoring old 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

Rehumanising system administration

- The challenge is now Knowledge
  - Planning
  - Disseminating
  - Quality assuring
  - Understanding what you made!
- Good models bring simplicity and agility
  - Based on atoms with few tethers
- Pedagogical and didactic skills return
  - Humanities students can play an increasing role

Cfengine's promise model is an atomic knowledge model

```
files:

  “/etc/shadow” → { “security@my.org” },

        comment => “standardize non LDAP passwords”,
    edit_lines => set_password(“root”,”xyz123”);

services:

 "Themes"
          service_policy => "start",
    service_dependencies => { "Alerter" },
         service_method => exmethod;
```

Free/Open Source?

- The source code is unimportant – social aspect is
  - Insight and shared experience (knowledge)
- Assurances for users – risk reduction
  - A lot of commercialization of FOSS – this is a sign that it is now considered valuable and mainstream, guarantees it a certain future
- To drive adoption of rational technology, it is helpful to have
  - Low adoption price
  - User communities
- Fan-bases and loyalties can hinder progress

Summary

Free adaptability, stability and complexity

Fewer dependencies + simple clear promises

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

Free Speech or Free Beer?