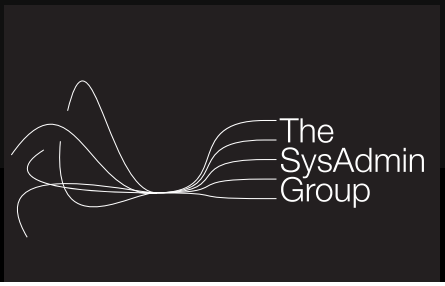
15 Years of DevOps

Geoff Halprin
The SysAdmin Group
geoff@sysadmin.com.au

LISA 2012 Conference
San Diego, December 2012



(C) Copyright 2012, Geoff Halprin.

We will all look a lot more like Google over the next decade.

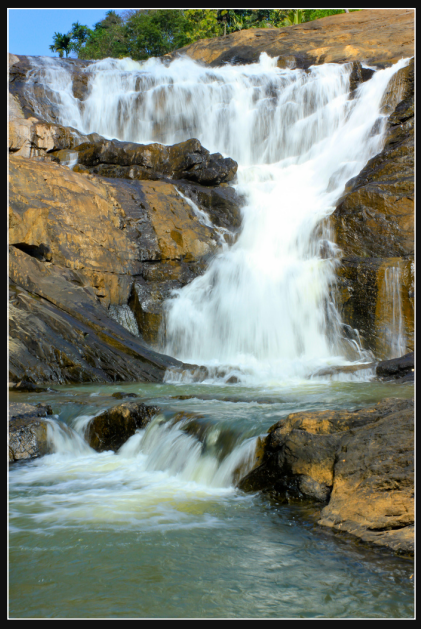
Software Development has changed forever.

Now it's System Administration's turn.

<1>

A Short History of Software Development

Software Development in the 80s



The Age of the Dinosaurs

http://www.flickr.com/photos/cblue98/6679374097/

The Waterfall Model

Highly Formalised, Unidirectional Workflow

Requirements	Architecture	Design	Code	Test	Release
Input: Requirements from user	**Input:** RDD	**Input:** SAD	**Input:** DDD	**Input:** RDD, SAD, DDD, Code	**Input:** Code, Test Reports
Output: Requirements Definition Document (RDD)	**Output:** System Architecture Description (SAD)	**Output:** Detailed Design Description (DDD)	**Output:** Code	**Output:** Test Reports	**Output:** Product

- **Key Attributes:**
  - Left-to-right progress. Each phase completed before next commenced (in theory).
  - Right-to-left traceability. Code can be traced back to requirements.

Lie #1: It was always a bad guess

Lie #2: The world does not stand still

Lie #3: Not all developers are equal

Lie #4: The A-team versus the B-team

The Big M Methodologies

The Root Causes

- **Human Resources**
  - They eschewed the difference in productivity of programmers.
- **Wrong Model**
  - All the creativity is at the beginning, the rest is just building.
- **High Risk, Hence High Formality**
  - Such large investments require significant controls to (attempt to) ensure that the value is delivered.

The Environment Was Changing, Too

Software was no longer delivered monolithically

- **Smaller, Interconnected Systems**
  - 2-Tier (Client/Server) networking model (RPC)
    - Thick clients
  - Enterprise Service Bus
- **Software Design Patterns (1994)**
- **The Web**
  - No longer creating long distribution chains for physical media.
  - 3-tier architecture (Model, View, Controller)

The Agile Manifesto (2001)

Putting the SDLC on its head

We Favour:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

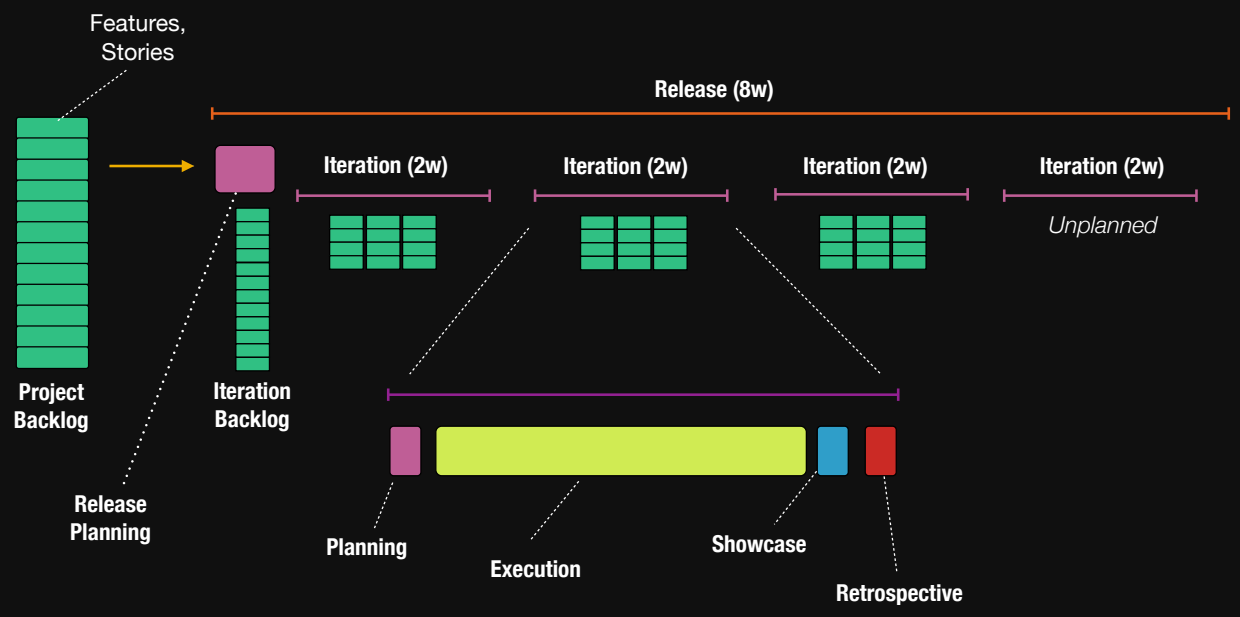
Agile Principles

We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximising the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organising teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

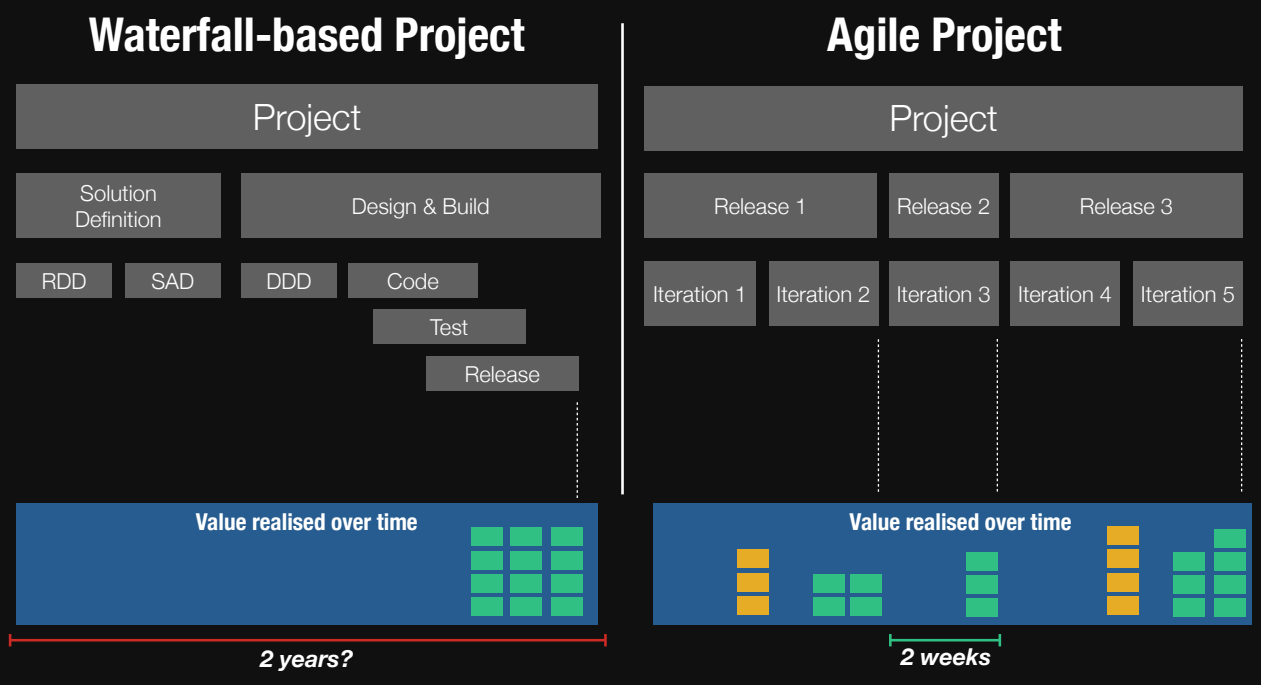
Project Cadence

New words, but simple concepts



Agile Versus Waterfall

Business Value Faster and More Reliably



Dev vs Ops, Part 1

A rivalry with much history

- **The First Wrong Assumption**
  - That Production looks just like Development.
- **The Second Wrong Assumption**
  - The Production looks just like a vanilla vendor install.
- **The Third Wrong Assumption**
  - That the Production environment is static.

How Did it Get This Way?

The Methodologies Assumed We Would Talk

- **ISO/IEC-12207:1995 SDLC**
  - Does not mention system operations at all
- **IEEE 1074-1991**
  - Does not mention system operations at all
- **Rational Unified Process (2003)**
  - 3½ whole pages on deployment!
- **They all assume that the programmers will deal with the operational aspects of the system as part of their normal SDLC activities.**
  - This has been proven to be a bad assumption!

The Great Assumption

All These Methodologies Assume We Are “In The Room”

- **This same mistake in thinking can derail Agile software development projects**
  - If your company is doing Agile software development, are Operations participating?
  - Are you placing “non-functional” stories in the backlog, or is the only ‘customer’ the one wanting new features?
- **DevOps puts an Ops person in the room**
  - That doesn’t guarantee the right stories are placed on the project backlog.

Serviceability Criteria

Making a System Administrator’s Life Less Stressful

- **Each production environment is unique.**
  - Applications must adapt to their environment.
- **The environment that an application is deployed into is never static across its life.**
  - Applications must cope with the world changing around them.
- **These goals are achieved by the application exposing the necessary** *controls* **to the system administrator to provide for maintenance.**
- **This is actually a definable set of criteria that, when met, make a system maintainable across its life.**
  - We will call these the ***serviceability criteria***.

Operational Requirements

aka “Non-Functional Requirements”

- **Serviceability Criteria (Scope: Global)**
  - Wouldn’t it be nice if all applications exposed these controls in a uniform manner?
- **Standard Operating Environment (Scope: Local)**
  - All applications within a particular environment share these. e.g. naming conventions, filesystem layout standards.
- **Application Non-Functional Requirements (Scope: Application)**
  - These are unique to each application. e.g. Application performance targets. Data storage requirements.

As an industry, we have left each individual sysadmin to fight for themselves (and repeat the same mistakes).

We need to do better.

The Grammar of Programming

One Last Thing...

- **All programming projects require the basics to be correct:**
  - Software Change Management
    - aka Change Control
  - Software Configuration Management
    - aka Version Control
  - Software Build and Release Management

</1>

<2>

A Short History of System Administration

An Apology

The following history is highly subjective and superficial. It ignores by omission the incredibly important contributions of a great many people.

This was not intentional.

Firefighters First and Foremost



Hardware was expensive

People were cheap

The Age of the Pioneer

You are in a maze of twisty passages, all different

- **Large Systems**
- **Many Networking Variants**
  - TCP/IP, SNA, X.25, Token Ring, DECnet, Netware
- **Bespoke Configurations**
- **Vertical Scaling**
- **Everyone had a Unix variant**

What's a System Administrator?

A well-kept secret!

- **LISA Conference: 1986**
  - 100 users or over 1GB of storage
- **SAGE: 1992 (SAGE-AU: 1993)**
  - Earlier: Sun User Group, etc.
- **What's in a name?**
  - System Programmer, System Administrator, Network Administrator, Network Engineer
  - Officially: Programmer, Research Scientist, ...

The Profession is Developing

LISA plays a key role in bringing thought leaders together

- **SA-CMM (1993)**
- **SAGE Job Descriptions Booklet (1993)**
- **Formal courses in SysAdmin at CQU (Aus) and U. Oslo**

The Network Isn't The Computer

But TCP/IP is here to stay

- **The Internet**
  - TCP/IP becomes the standard network.
- **Custom IT Solutions**
  - Everyone is special. Every solution unique.
- **Tools begin to emerge:**
  - sudo (1980), top (1984), rdist (1986), perl 4.036 (1993), cfengine (1993), mrtg (1995), ssh (1995), rsync (1996)

The SA-BOK

Before ITIL, there was SA-BOK

<table>
<tr><th>To Control</th><th>To Organise</th><th>To Protect</th><th>To Optimise</th><th>To Plan</th></tr>
<tr><td>Change Management</td><td>Facilities Management</td><td rowspan="2">Data Security</td><td rowspan="2">Performance Management</td><td rowspan="2">Capacity Planning</td></tr>
<tr><td rowspan="2">Problem Management</td><td>Network Management</td></tr>
<tr><td rowspan="3">Server Management</td><td rowspan="3">Business Continuity Planning</td><td rowspan="3">Process Automation</td><td rowspan="3">Technology Planning</td></tr>
<tr><td rowspan="2">Production Management</td></tr>
<tr><td>Software Management</td></tr>
<tr><td>Asset Management</td><td>Data Management</td></tr>
<tr><td colspan="5">Service Management</td></tr>
</table>

The Rise of The Web

With great power comes great responsibility

- **The Emergence of The Web**
  - Thin clients!
- **Three Tier Infrastructure Model (Web-App-DB)**
  - Physical Servers Rule
- **Linux at the Edge; Solaris at the Core**

The Commoditisation of IT

Penguin rising

- **Commodity Hardware**
  - Sun, IBM, Intel still battling for the enterprise.
  - Intel/Linux has won the startups.
- **The Rise of Virtualisation**
  - VMware and Linux drive adoption of x86 architecture.

The Emergence of the Cloud

Nothing will ever be the same again

- **The Wars are over: Linux, x86, VMs.**
  - Amazon Web Services. Infrastructure on demand.
- **The Rise of the API**
  - Infrastructure as code.
- **Automation driven by scale**
  - The WebOps Movement.
  - The DevOps Movement.
- **What's in a name (part 2)?**
  - SRE, Web Ops, DevOps...

</2>

<3>

The Rise of DevOps

The same way that failure of Waterfall drove developers to change the way we developed software, Cloud-based sites and the need for rapidly provisioned, highly scalable infrastructure drove the way we managed infrastructure.

What is DevOps?

DevOps is to System Administration what
Agile is to Software Development

- **Culture and Attitude**
  - First and foremost, DevOps is about creating a collaborative environment between Dev and Ops, integrating the teams as much as possible.
- **Practices and Processes**
  - Automate as much as possible.
- **Technology and Tools**
  - Old and New tools to support the effort.

DevOps Culture

It all starts by sitting on the same side of the table

- **The simple act of colocating Dev and Ops people is the single, largest step.**
  - Visibility of each others work through Kanban walls.
  - Shared food and discussions.
  - Cross-over of teams.

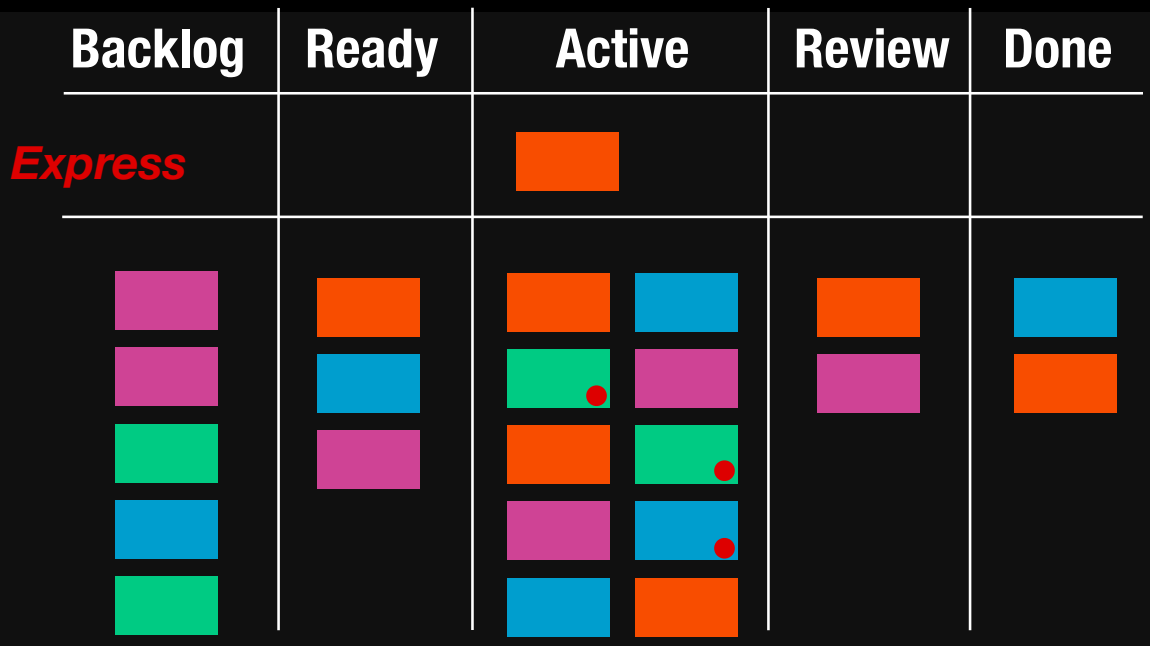
DevOps Practices

Some common DevOps practices

- **Developer Behaviour Changes**
  - Infrastructure as Code
  - Continuous Integration
  - Production Support (carrying a pager!)
- **SysAdmin Behaviour Changes**
  - Embedded with Dev teams
  - Kanban walls for Ops

Kanban Walls for Ops

Visibility is the first step to collaboration



DevOps Tools

So many (these are just a few)

- **Configuration Management**
  - Puppet, Chef, Babushka, cfengine
- **Software Change Management (Version Control)**
  - Github
- **Continuous Integration**
  - Jenkins, Capistrano
- **Testing**
  - Rails, Cucumber, Vagrant

And Frameworks

Infrastructure in the Cloud

- **Commercial**
  - Amazon Web Services
  - VMware Dynamic Ops, BMC CLM
- **Open Source**
  - Open Stack, CloudForms, Eucalyptus, Open Nebula, Open Cloud Framework,
  - Cloud Foundry, AppFog, Heroku, OpenShift
- **Standards**
  - ???

Cucumber

Adapting Agile tools to DevOps

```
  # language: en
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario Outline: Add two numbers
    Given I have entered <input_1> into the calculator
    And I have entered <input_2> into the calculator
    When I press <button>
    Then the result should be <output> on the screen

  Examples:
    | input_1 | input_2 | button | output |
    | 20      | 30      | add    | 50     |
    | 2       | 5       | add    | 7      |
    | 0       | 40      | add    | 40     |
```

Babushka

Test-Driven System Administration

```
dep 'ruby 1.9.2 in use' do {

met? {
  shell('ruby -version')['ruby 1.9.2 p0']
}
meet {
  shell('rvm use 1.9.2')
}

end
```

Idempotent Scripting!

What DevOps Does

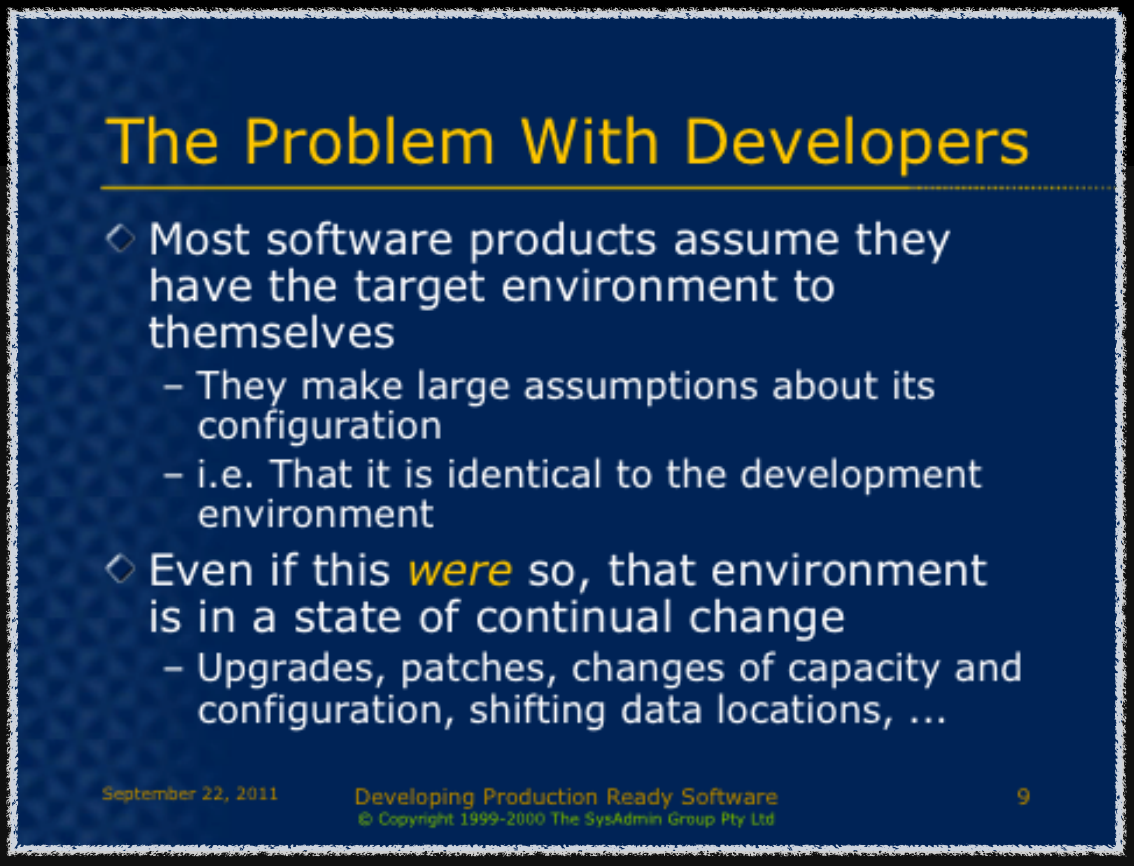
Breaks down the silos

- **Ops are now "in the room" with the Dev team.**
  - Dev do after hours support!
  - Dev now build more maintainable code.
- **An end-to-end system view**
  - Dev now see (and respond to) the problems their code causes across the entire lifecycle.
- **Configuration Management is automated.**
- **Change Management is automated.**
- **Continuous Integration and Release**

Continuous Integration and Release

Keeping the Code Clean

- **Continuous Integration is a Development Team Practice**
  - It requires well-behaved developers and a clean code base.
- **Continuous Release is a DevOps Practice**
  - It requires a well-defined (automated) release procedure.
  - It requires well-defined target infrastructure.



What DevOps Isn't

Fully cooked...

- **A silver bullet**
  - Systems still break. Patches still need to be applied.
  - Not all products fit a pure DevOps model.
- **Complete or holistic**
  - ITIL? SA-BOK? Normal lifecycle?
- **Highly scalable into traditional enterprise environments**
  - Agile started in small teams. It took a while to learn how to scale it.
  - DevOps has started with highly specialised applications, developed to leverage the DevOps model.

Problems DevOps Doesn't Solve

DevOps sees the vertical world of an application

- **Serviceability Criteria**
- **Technical Debt / Shoemaker Projects**
- **Service Monitoring**
- **The Rest of the Ops Lifecycle**
- **Professional Ethics, Training, etc.**

Web Ops versus Enterprise Ops

Does DevOps Fit the Enterprise?

- **You can’t just throw away infrastructure**
- **Vertically scaled apps**
- **Enterprise Change Management**
- **Poorly behaving vendors**

Badly Written Software

Vendors have been writing bad software for a very long time...

- 1994: No, you can't have root (Craig Bishop)
- 1995: Guidelines for software developers (Geoff Halprin)
- 2001: The problem with developers (Geoff Halprin)
- 2005: Software release engineering (Geoff Halprin and Lee Damon)

The Problem

- There is a problem out there:
  - Developers (still) don't develop maintainable, production-ready, manageable code.
  - This failure must then be handled by system administrators who must second-guess the developers, both as to their intent and the behaviour of the implemented system.
  - *This is not acceptable.*

© Copyright 1999-2000 The SysAdmin Group Pty Ltd

Some Examples

- 70KB /bin/sh script that must run as *root*
- Microsoft DLLs planted in the WIN32 directory
- Licence file that must be in a particular i-node [Interleaf]
- Install scripts that expect to extract directly from one type of media (i.e. "mt fsf 2") [Orbix]
- Systems that require a graphical installer

© Copyright 1999-2000 The SysAdmin Group Pty Ltd

Some More Examples

- Packages with force interactive installation (Solaris *pkginstall* with tty read)
- Packages that require an encryption/ activation key to install, not just to activate.
- Systems that insist on being installed in one location
- Worse: those that install in other locations, but don't work there! [Palm Desktop]

© Copyright 1999-2000 The SysAdmin Group Pty Ltd

</3>

<4>

DevOps Isn’t New

System Administration

A definition

- To maintain the availability and integrity of computing resources
- To support and encourage the effective use of those resources

Our Motivation

To make ourselves redundant

- People don't like doing work
- Lazy people invent ways to avoid work
- Therefore, all progress is made by lazy people



Making Ourselves Redundant

- Strategy 1: Consistency
  - More consistency = reduced effort across a fleet
- Strategy 2: Autonomous Systems
  - The more they look after themselves, the less for us to do!
- Strategy 3: Working on the Right Problems!
  - Automate the tedium.
  - Capture your expertise.

Changing System Files

Working on the right problem

- **Our job should not be a memory test**
- **We should concentrate on where the best payoff is**
- **Why do I have to remember which signal to send to which process for each configuration file?**
  - Or which ones are binary and need a different editor?

Change

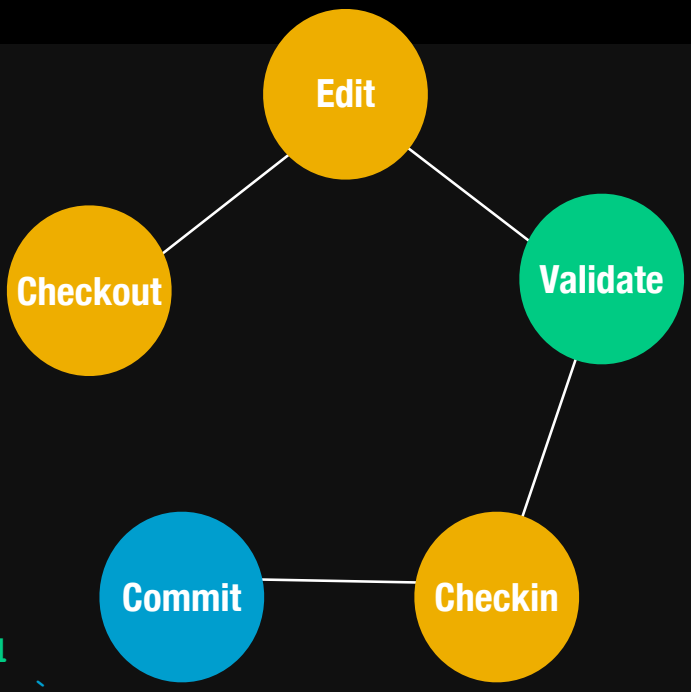
Why not edit all files the same way?

```
# change /etc/passwd

/etc/.change:
Default:
  checkout rcs co %%
  checkin rcs ci %%
  edit vi %%
end

passwd:
  edit: vipw
end

inetd.conf:
  validate: v_etc_inetd
  commit: kill -HUP `...`
```



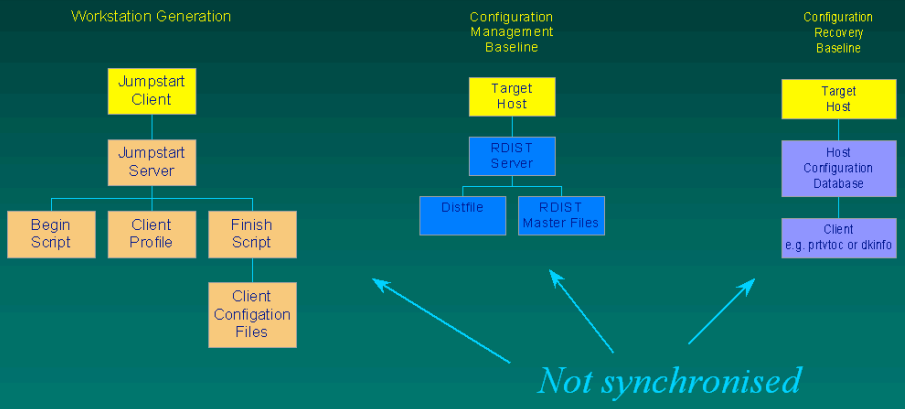
Structured Systems Administration

Moving from managing a single host to managing a fleet

- Treat the whole network as a single system; not as a large number of independent hosts.
- Consistency
- Scalability

Traditional Systems Management



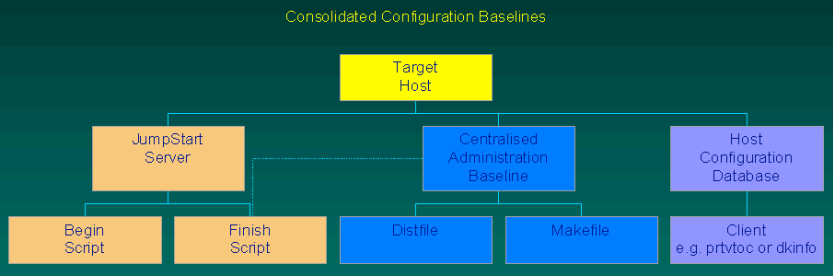


You are in a maze of twisty passages, all different

(C) Copyright 1996-1997, The SysAdmin Group Pty. Ltd.

A Better Way

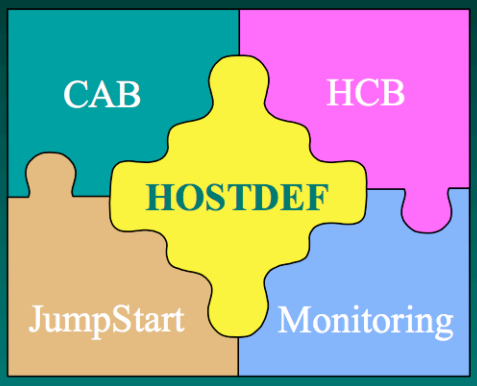




You are in a maze of twisty passages, all alike

(C) Copyright 1996-1997, The SysAdmin Group Pty. Ltd.

Structured Systems Management Key Components



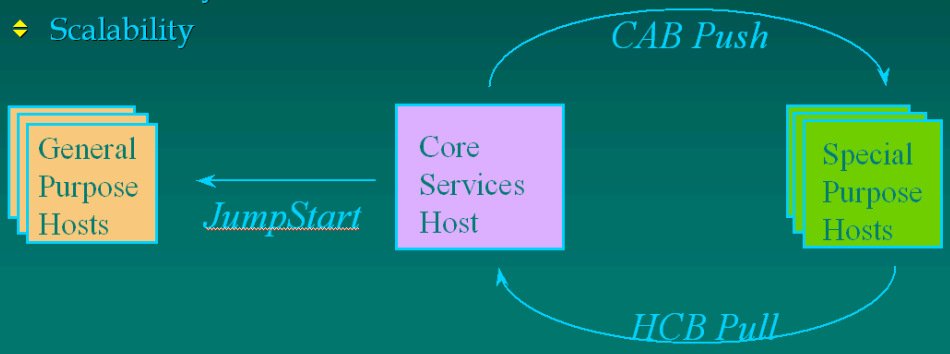
- CAB
- HCB
- JumpStart
- Monitoring
- HOSTDEF

Rigidly defined areas of doubt and uncertainty

(C) Copyright 1996-1997, The SysAdmin Group Pty. Ltd.

Structured Systems Management

- Treat the network as a single system; not as a large number of independent hosts.
- Consistency
- Scalability



(C) Copyright 1996-1997, The SysAdmin Group Pty. Ltd.

The File Distribution Process

- CAB is a *push* distribution system
- Underlying mechanism is **rdist 6.1.0**
- Flexible management through combining **make**, **rdist** and **sdist** (**rdist** over **ssh**).
- **Make** ensures the correctness of each functional area.
- **Rdist** distributes packages to various host classes as appropriate.

(C) Copyright 1996-1997, The SysAdmin Group Pty. Ltd.

Maintaining the Baseline

- PUSH.hostdef
  - Defines each host's characteristics;

```
HOST btsw001
        site=bt
        os=solaris2
        client
ENDHOST
```

- PUSH.rules
  - Determines which hosts are placed in each host class.

```
BTHOSTS      ( $h{site} eq "bt" )
BTS2HOSTS    ( $h{site} eq "bt" && $h{os} eq "solaris2" )
BRSW         ( $h{site} eq "br" && $h{client} )
BTSW         ( $h{site} eq "bt" && $h{client} )
```


(C) Copyright 1996-1997, The SysAdmin Group Pty. Ltd.

Structured Systems Management

The Philosophy

- Treat the network as a single system; not as a large number of loosely-coupled independent hosts.
  - Centralised Configuration Management
  - Centralised Host Administration
  - Centralised Host Monitoring
- Uniform Framework Deployed Across All Hosts
  - Common Framework
  - Known Points of Flexibility

1995



Rigidly defined areas of doubt and uncertainty

ISP in a Box

Spinning up a Point-of-Presence in 30 minutes
(after the physical rack-and-stack)

- **Components**
  - Configurator. Build configuration files from a central master properties file.
  - Jumpstart with a whole bunch of post-provisioning, to install patches (with reboots), software, configuration, etc.
  - Configuration of Cisco routers.
  - Configuration of central servers to talk to the new POP.
- **This was all done in 1997**

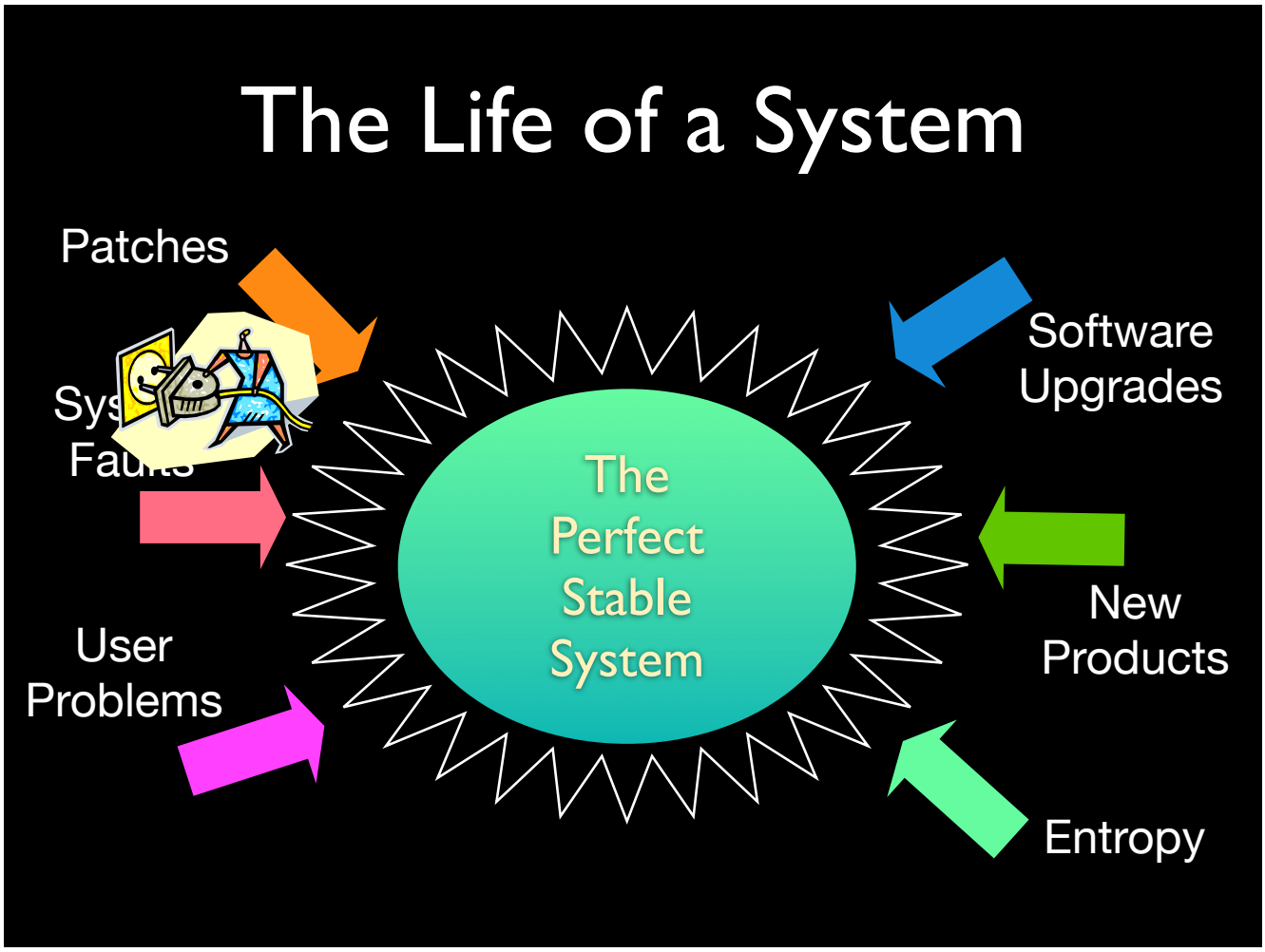
</4>

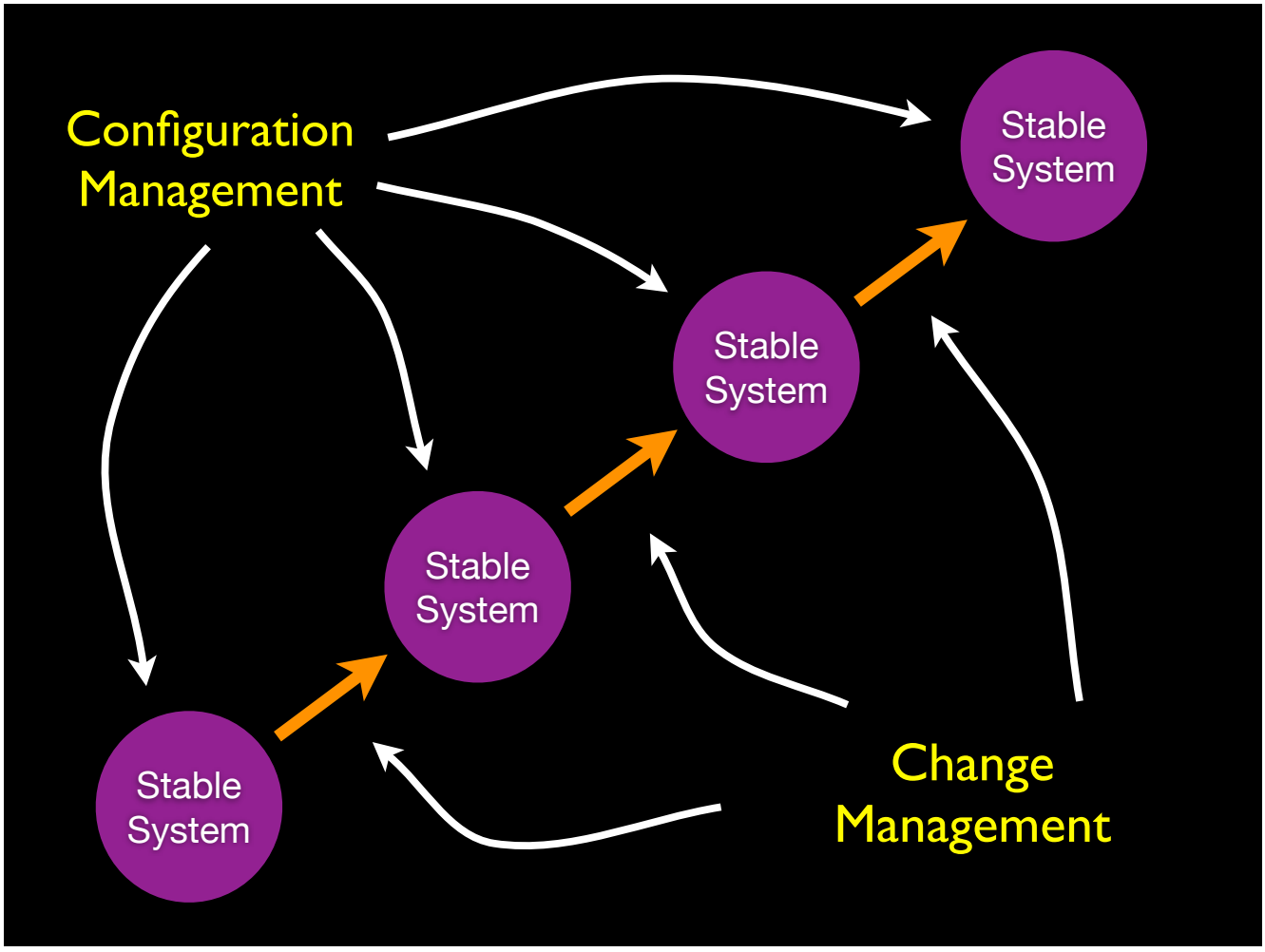
<5>

Let’s look a little deeper at some System Administration practices...

Change Management

The Life of a System





The System as a State Machine

- **Configuration Management**
  - Management of State Definition and Integrity
- **Change Management**
  - Management of State Transition Integrity

Configuration Management

- **Managing State Integrity**
  - Ability to accurately define system components (CMDB)
  - Ability to accurately define system state (configuration files, etc)
  - Ability to recover system state (backups)

Change Management

- **Managing State Transition Integrity**
  - Correctness of target state
  - Correctness of work instructions

Change Management = Risk Management

The Risks Being Managed

The Purpose of Change Management is to manage the risks associate with state transition

- **Process Failures**
  - Incorrect Change Procedure
  - Procedure Not Followed Correctly
  - Unexpected Side-Effect
- **End State Failures**
  - Toxic End State

Risk Mitigation

All Changes share the same risk mitigation strategies

- **Backups**
  - The point in time directly before the change is the last known stable point, hence the baseline for all other mitigation plans.
- **Other techniques**
  - Copy files sideways. Split mirrors. Snapshots.
  - Change qualification. Progressive deployment.
  - Copilots. Contingency time.
  - Automation.

DevOps and Change Management

DevOps sees everything as a code push

- **DevOps is highly optimised for a single type of Change -- the code push.**
  - This often requires a full build from scratch.
  - They also deal with complex changes, like database schema changes with minimal service disruption.
  - Service availability is paramount in DevOps change philosophy.

Configuration Management

Traditional Configuration Management

System Administration has been talking about Configuration Management for a long time!

- **Many Tools**
  - rdist, cfengine, bcfg2, satan, Netomata, puppet, chef, ...
- **Many Approaches**
  - push/pull, centralised/distributed, various abstraction layers and models.
- **Many Papers**
  - Dozens of papers over 26 years.

DevOps and Configuration Management

DevOps is highly reliant on automated Configuration Management

- **DevOps solves this for individual applications.**
  - The vertical stack.
  - Often by blowing a VM away and starting again.
- **Not the complexity of the entire infrastructure.**
  - Assumes shared infrastructure (physical compute, store, network) are all in place (and configured!).

DevOps doesn’t solve these problems, but it does establish a better culture for solving them!

</5>

<6>

DevOps is the continuation of the path we have been on since the beginning.

DevOps is filling the void that our lack of progress as a profession has left unanswered.

Where does that path lead?

The Software Defined Data Centre

Everything is software now

- **Virtualisation is only the first step**
  - It is necessary, but not sufficient
  - 20K machines virtualised are still 20K machines individually managed.
- **Automation is critical to that journey**
  - Standardisation is the basis for automation.
- **Cloud standards are in their infancy**
  - Expect a lot of change over the coming years.

The Automation Manifesto

Realising the Cloud

We Favour:

VM templates and golden images over **Standards and documents**

Automated processes over **Documented procedures and build instructions**

Self-service of standardised offerings over **Custom designs**

Consistency of hosts over **Custom configurations**

Machine parseable hand-off artefacts over **Human written and processed documents**

Low risk "standard" change activities over **Complex, scheduled change windows**

Centralised management platforms over **Individual host and component management**

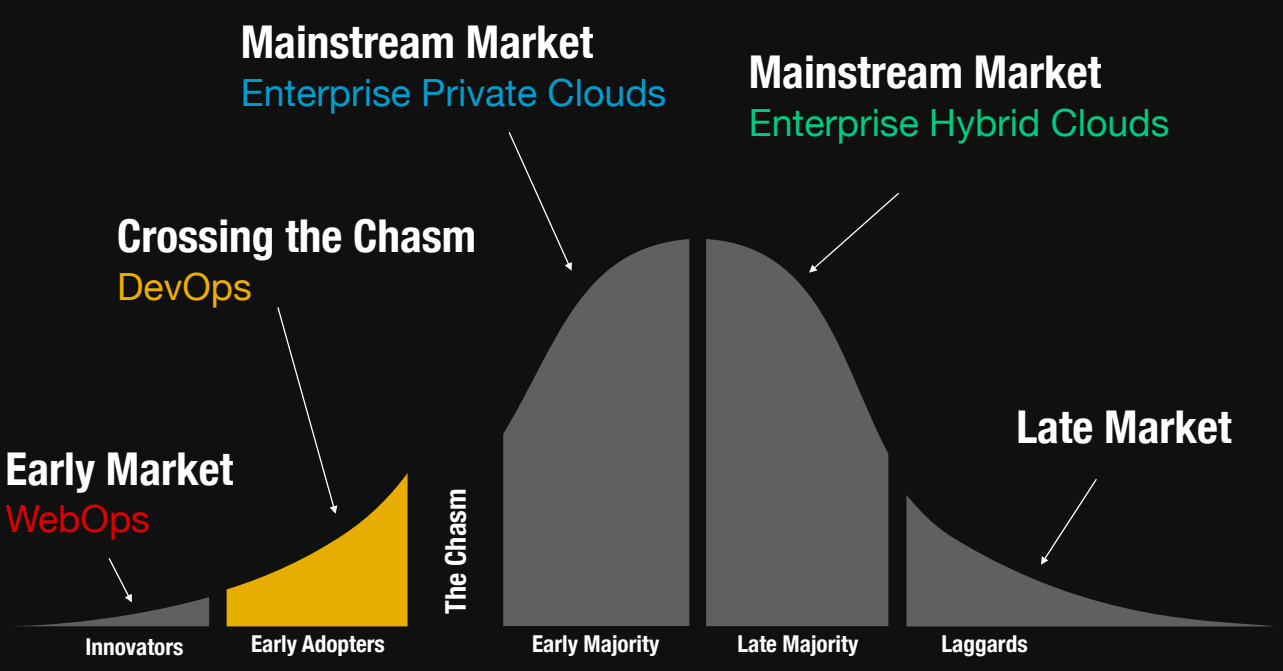
Industry standards over **Proprietary protocols**

Consistency of toolset over **Consolidation of products**

© Geoff Halprin

DevOps and Automated Infrastructure

DevOps is the harbinger of the next wave



Infrastructure delivered via API will be the basis for large scale enterprise outsourcing of infrastructure and associated support teams.

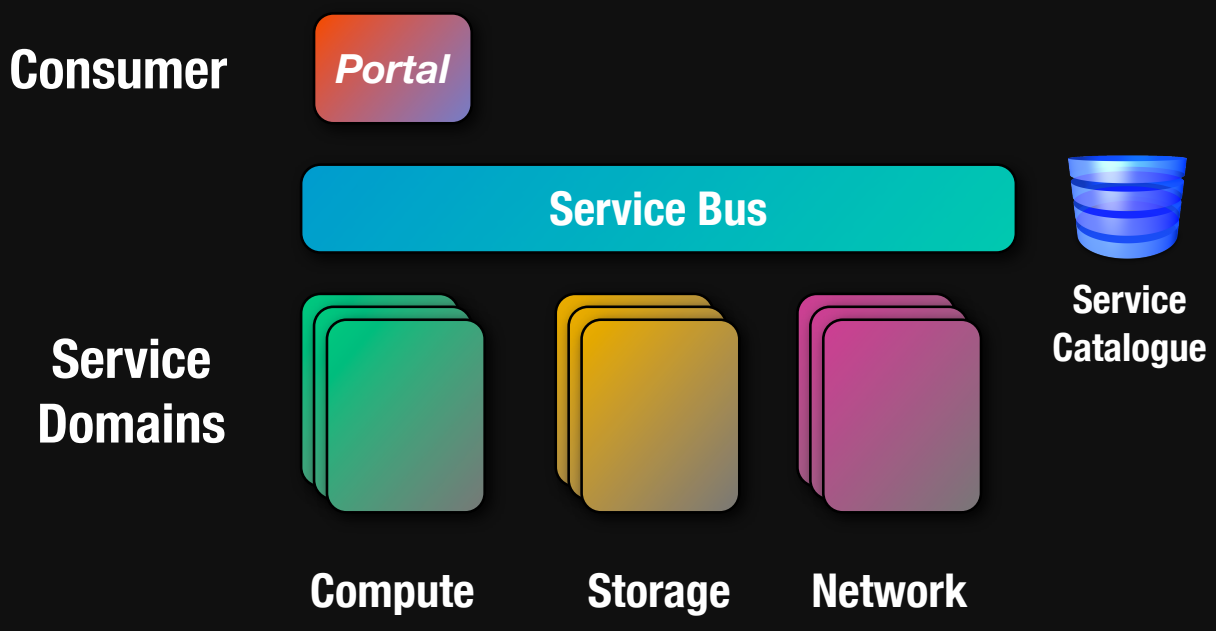
The Next Outsourcing Wave?

Infrastructure delivered via API will be the basis for large scale enterprise outsourcing of system teams

- **Rob Thomsett: Process versus Project Work**
  - First generation of outsourcing was just replacing staff.
  - Infrastructure was bespoke; each machine managed as an individual, customised solution.
- **Standardised infrastructure delivered by API**
  - You can shop around; you can use multiple providers simultaneously.

The Infrastructure Service Bus

A market place for services



The Infrastructure Service Bus

This really will change the industry

- **Enterprise (Infrastructure) Service Bus**
  - Service Provider Model
  - Being embraced by large enterprises now!
  - The outsource providers are becoming cloud providers!
- **Our Cloud APIs are evolving into Middleware (message bus) APIs**
  - System Administrations looks even more like enterprise software systems!

The irony here is that as infrastructure is treated more like code, the very frameworks and patterns used for software development become applicable to infrastructure.

</6>

<7>

- **The cost models of Enterprise IT are broken**
  - If you don't move to a cloud model for IT service delivery, you will be outsourced.

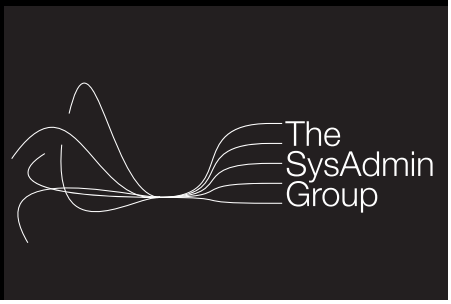
- **We are now in the decade of the API**
  - *Everything* is code.

- **We are at the beginning of this transition**
  - There's plenty for everyone to do!

What you do next is up to you...

The End

Comments to:
geoff@sysadmin.com.au



References

Some of the source data used in this talk

- **The Agile Manifesto**
  - http://www.agilemanifesto.org
- **The Facts and Fallacies of Software Engineering**
  - Robert L. Glass, ISBN 0-321-11742-5.
- **System Administrator's Body of Knowledge (SA-BOK)**
  - http://www.sysadmin.com.au/sa-bok.html
- **Serviceability Criteria**
  - https://www.sysadmin.com.au/whitepapers/developer_guidelines-0212.pdf