Comprehensive Anticipatory Systems Administration

Thoughts About Eternally Regenerative Software Administration

CJ Fearnley

LinuxForce, Inc.
http://www.LinuxForce.net

3 September 2008
Presentation to PLUG
Philadelphia Area Linux User's Group

`http://www.CJFearnley.com/comprehensive.anticipatory.SA.pdf`

Outline

My Background

- In 1995, about 13 years ago, I started doing professional systems administration work
- About the same time I helped start PLUG
- In addition to my day job as President and CEO of LinuxForce, I am Executive Director of the Synergetics Collaborative, a non-profit building on Buckminster Fuller's Synergetics.
- In 1994, I wrote and put on-line the first version of The R. Buckminster Fuller FAQ
- In 1991, **Trimtab**, the newsletter of the Buckminster Fuller Institute published my essay Reading Synergetics: Some Tips which is on-line at `http://www.cjfearnley.com/synergetics.essay.html`

My Background

- In 1995, about 13 years ago, I started doing professional systems administration work
- About the same time I helped start PLUG
- In addition to my day job as President and CEO of LinuxForce, I am Executive Director of the Synergetics Collaborative, a non-profit building on Buckminster Fuller's Synergetics.
- In 1994, I wrote and put on-line the first version of The R. Buckminster Fuller FAQ
- In 1991, **Trimtab**, the newsletter of the Buckminster Fuller Institute published my essay Reading Synergetics: Some Tips which is on-line at `http://www.cjfearnley.com/synergetics.essay.html`

My Background

- In 1995, about 13 years ago, I started doing professional systems administration work
- About the same time I helped start PLUG
- In addition to my day job as President and CEO of LinuxForce, I am Executive Director of the Synergetics Collaborative, a non-profit building on Buckminster Fuller's Synergetics.
- In 1994, I wrote and put on-line the first version of The R. Buckminster Fuller FAQ
- In 1991, **Trimtab**, the newsletter of the Buckminster Fuller Institute published my essay Reading Synergetics: Some Tips which is on-line at `http://www.cjfearnley.com/synergetics.essay.html`

My Background

- In 1995, about 13 years ago, I started doing professional systems administration work
- About the same time I helped start PLUG
- In addition to my day job as President and CEO of LinuxForce, I am Executive Director of the Synergetics Collaborative, a non-profit building on Buckminster Fuller's Synergetics.
- In 1994, I wrote and put on-line the first version of The R. Buckminster Fuller FAQ
- In 1991, **Trimtab**, the newsletter of the Buckminster Fuller Institute published my essay Reading Synergetics: Some Tips which is on-line at `http://www.cjfearnley.com/synergetics.essay.html`

My Background

- In 1995, about 13 years ago, I started doing professional systems administration work
- About the same time I helped start PLUG
- In addition to my day job as President and CEO of LinuxForce, I am Executive Director of the Synergetics Collaborative, a non-profit building on Buckminster Fuller's Synergetics.
- In 1994, I wrote and put on-line the first version of The R. Buckminster Fuller FAQ
- In 1991, **Trimtab**, the newsletter of the Buckminster Fuller Institute published my essay Reading Synergetics: Some Tips which is on-line at `http://www.cjfearnley.com/synergetics.essay.html`

Who Was Buckminster Fuller?

Richard Buckminster “Bucky” Fuller (July 12, 1895 - July 1, 1983) was an American architect, author, designer, futurist, inventor, and visionary.

— Wikipedia

What is Systems Administration

- Systems Administration is caregiving for computer systems.

Notable Quotable

The major cause of problems are solutions.

— Eric Severeid

I call this the plight of the systems administrator

What Does Buckminster Fuller's Synergetics have to do with Systems Administration?

What Does Buckminster Fuller's Synergetics have to do with Systems Administration?

That question has been explicit in my mind but only implicit in my work for most of the last 13 years. I will try to disclose the connection in the remainder of the talk.

In short, comprehensive anticipatory systems administration is my interpretation of Buckminster Fuller's engineering philosophy (which Fuller branded in several partially overlapping ways: **Synergetics**, **Comprehensive Anticipatory Design Science**, **The World Game**, and other supporting phrases such as **"doing more with less"**, etc) applied to computer systems management.

What Does Buckminster Fuller's Synergetics have to do with Systems Administration?

What Does Buckminster Fuller's Synergetics have to do with Systems Administration?
That question has been explicit in my mind but only implicit in my work for most of the last 13 years. I will try to disclose the connection in the remainder of the talk.

In short, comprehensive anticipatory systems administration is my interpretation of Buckminster Fuller's engineering philosophy (which Fuller branded in several partially overlapping ways: **Synergetics**, **Comprehensive Anticipatory Design Science**, **The World Game**, and other supporting phrases such as **"doing more with less"**, etc) applied to computer systems management.

What Does Buckminster Fuller's Synergetics have to do with Systems Administration?

What Does Buckminster Fuller's Synergetics have to do with Systems Administration?
That question has been explicit in my mind but only implicit in my work for most of the last 13 years. I will try to disclose the connection in the remainder of the talk.

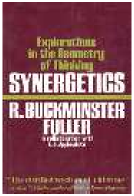
In short, comprehensive anticipatory systems administration is my interpretation of Buckminster Fuller's engineering philosophy (which Fuller branded in several partially overlapping ways: **Synergetics**, **Comprehensive Anticipatory Design Science**, **The World Game**, and other supporting phrases such as **"doing more with less"**, etc) applied to computer systems management.

Notable Quotable

Dare to be Naïve!

— Buckminster Fuller

Comprehensive Thinking



Synergetics is the system of comprehensive thinking which R. Buckminster Fuller introduced and began to formulate (primarily, in his two volume magnum opus, *Synergetics: Explorations in the Geometry of Thinking*, 1975, 1979).

A mathematical model of how the world *really* works

Synergetics is an attempt to build a realistic, comprehensive mathematical modeling system (a coordinate system: think: to coordinate "everything" comprehensively) to help guide Humanity to better understand our Universe and to achieve success ("to make the world work for 100% of humanity in the shortest possible time, with spontaneous cooperation and without ecological damage or disadvantage of anyone" — R. Buckminster Fuller).

Fuller's system of epistemography and mathematical-physics attempts to disclose how Nature actually operates — her "operational mathematics."

Humanistic and Philosophical

"Nature's coordinate system is called Synergetics—synergy means behavior of whole systems unpredicted by any part of the system as considered only separately. The eternally regenerative Universe is synergetic. Humans have been included in this cosmic design as local Universe information-gatherers and local problem-solvers in support of the integrity of the eternal, 100-percent-efficient, self-regenerative system of Universe. In support of their cosmic functioning humans were given their minds with which to discover and employ the generalized laws governing all physical and metaphysical, omniinteraccommodative, ceaseless intertransformings of Universe."

— Buckminster Fuller

Teleology, Problem-Solving, & Design

Synergetics is teleological (the philosophical study of design and purpose): it provides a comprehensive, anticipatory design science method and philosophy for understanding Universe and its applications to problem-solving and design in all areas of human endeavor.

Complex and Multi-Faceted

Synergetics is multi-faceted: it involves geometric modeling, scientific discipline in examining the facts of experience from a fresh perspective, exploring their inter-relationships, and the process of thinking. Synergetics endeavors to identify and understand the principles that Nature actually employs in coordinating Universe (both physically and metaphysically).

The Study of Spatial Complexity

"Synergetics, in the broadest terms, is the study of spatial complexity, and as such is an inherently comprehensive discipline. ... Experience with synergetics encourages a new way of approaching and solving problems. Its emphasis on visual and spatial phenomena combined with Fuller's wholistic approach fosters the kind of lateral thinking which so often leads to creative breakthroughs."

– Amy Edmondson, A Fuller Explanation: The Synergetic Geometry of R. Buckminster Fuller, 1987

Synergetics is Integrative and Experiential

- Synergetics is *integrative*: it seeks relationships between systems and events (in contradistinction to analysis which breaks systems into parts)
- Synergetics is *experiential*: Synergetics Dictionary, Vol. 4, p. 91: "The difference between synergetics and conventional mathematics is that it is derived from experience and is always considerate of experience, whereas conventional mathematics is based upon 'axioms' that were imaginatively conceived and inconsiderate of information progressively harvested through microscopes, telescopes and electronic probings into the non-sensorially tunable ranges of the electromagnetic spectrum."

Synergetics is Integrative and Experiential

- Synergetics is *integrative*: it seeks relationships between systems and events (in contradistinction to analysis which breaks systems into parts)
- Synergetics is *experiential*: Synergetics Dictionary, Vol. 4, p. 91: "The difference between synergetics and conventional mathematics is that it is derived from experience and is always considerate of experience, whereas conventional mathematics is based upon 'axioms' that were imaginatively conceived and inconsiderate of information progressively harvested through microscopes, telescopes and electronic probings into the non-sensorially tunable ranges of the electromagnetic spectrum."

The Importance of Synergetics to Human Survival

“Most simply put, Synergetics is the study of how nature works, of the patterns inherent in nature, the geometry of environmental forces that impact on humanity. In his thousands of lectures, Fuller urged his audiences to study synergetics, saying ‘**I am confident that humanity’s survival depends on all of our willingness to comprehend feelingly the way nature works.**’”

Cheryl Lirette Clark, Ph.D.
12° of Freedom, Ph.D. Thesis
http://www.doinglife.com/12FreedomPDFs/Ib_AbstractLitReview.pdf

The Exposition of Synergetics: A Discovery Process

"Synergetics is the study of the way nature, physics and the universe works; it is '*the geometry of thinking*.' All of Fuller's work is an exposition of Synergetics following the development of his thinking as it evolved through experiments and practical applications in his artifacts. He says, '*The omnirational coordinate system which I have named synergetics is not an invention, it is purely discovery.*'"

Cheryl Lirette Clark, Ph.D.
12° of Freedom, Ph.D. Thesis
http://www.doinglife.com/12FreedomPDFs/Ib_AbstractLitReview.pdf

Alive and Evolving

Synergetics is alive and evolving in the minds and the work of the many people building on Bucky's work.

The **Synergetics Collaborative** is a center for education, collaboration, and development of Synergetics.
`http://www.Synergeticists.org`

Building Upon Bucky's Methods

As with any comprehensive system, Synergetics is incomplete. Bucky gave us a unique insight into what, where & how to explore.

The **Synergetics Collaborative** is working to build on Bucky's Synergetics to expand the work so that it can more effectively fulfill his vision.
`http://www.Synergeticists.org`

Resources

- Both volumes of *Synergetics: Explorations in the Geometry of Thinking* are on-line at
  `http://www.rwgrayprojects.com/synergetics/synergetics.html`
- Amy Edmondson's book *A Fuller Explanation: The Synergetic Geometry of R. Buckminster Fuller*
  `http://www.angelfire.com/mt/marksomers/40.html`
- *Reading Synergetics: Some Tips*
  `http://www.cjfearnley.com/synergetics.essay.html`
- Through Sep 21, The Whitney Museum in NYC has a major retrospective of Fuller's work: see http://www.whitney.org/www/buckminster_fuller/about.jsp

Resources

- Both volumes of *Synergetics: Explorations in the Geometry of Thinking* are on-line at

  `http://www.rwgrayprojects.com/synergetics/synergetics.html`
- Amy Edmondson's book *A Fuller Explanation: The Synergetic Geometry of R. Buckminster Fuller*

  `http://www.angelfire.com/mt/marksomers/40.html`
- *Reading Synergetics: Some Tips*

  `http://www.cjfearnley.com/synergetics.essay.html`
- Through Sep 21, The Whitney Museum in NYC has a major retrospective of Fuller's work: see http://www.whitney.org/www/buckminster_fuller/about.jsp

Notable Quotable

The most important fact about Spaceship Earth:

— Buckminster Fuller

Notable Quotable

The most important fact about Spaceship Earth: an instruction manual didn't come with it.

— Buckminster Fuller

Notable Quotable

[Design Science is] the effective application of the principles of science to the conscious design of our total environment in order to help make the Earth's finite resources meet the needs of all humanity without disrupting the ecological processes of the planet

— R. Buckminster Fuller

The Comprehensive Anticipatory Design Science Revolution

The Comprehensive Anticipatory Design Science Revolution designates the thread in Buckminster Fuller's philosophy that emphasizes the need to radically re-orient Humanity's problem-solving efforts by the employment of the practice of science in a comprehensive and anticipatory fashion toward the design of effective solutions to better meet the needs of 100% of Humanity. The emphasis is on the design of comprehensive solutions ... ones that work sustainably because they have anticipated comprehensively all factors relevant to the problem space.

Notable Quotable

There is only one revolution tolerable to all men, all societies, all political systems: revolution by design and invention.

— R. Buckminster Fuller

Notable Quotable

The success of all humanity can be accomplished only by a terrestrially comprehensive, technologically competent, design revolution. This revolution must develop artifacts where energy-use efficiency not only occasions the artifacts' spontaneous adoption by humanity, but also occasions the inadvertent, unregretted abandonment and permanent obsolescence of socially and economically undesirable viewpoints, customs and practices.

— R. Buckminster Fuller

Notable Quotable

The essential message of Fuller's design science is that human beings have access to the design laws of Universe, and a responsibility to use the extraordinary phenomenon of mind to discover and apply such principles. Our function is problem-solving. Synergetics, the discipline behind Fuller's more-with-less philosophy, above all encourages us to experiment. This material is superbly suited to nurture and enhance creativity, demanding both numerical rigor and intuitive leaps. ...
A design science revolution is imperative.

– Amy Edmondson

Notable Quotable

You never change things by fighting the existing reality. To change something, build a new model that makes the existing model obsolete.

— R. Buckminster Fuller

Notable Quotable

You can't better the world by simply talking to it.
Philosophy to be effective must be mechanically applied.

— Buckminster Fuller

FOSS: Free and Open Source Software

FOSS is software which is liberally licensed to grant the right of users to study, change, and improve its design through the availability of its source code.

— Wikipedia

FOSS and Humanity's Cultural Heritage

FOSS, seen as a growing part of Humanity's culteral heritage, is becoming an established resource for more and more of the IT infrastructure at more and more organizations.

Human society expands by building an inventory of tools created yesterday (often by the deceased) as a contribution to the next generation in an organic process of technological evolution. FOSS gives us maximum access to the code building this heritage.

Linux IS the Design Science Revolution Incarnate

Linux's history shows that it is a true instance of Fuller's Design Science Revolution. Notably, it is technology started by individual initiative (Linus Torvalds). Big Business, Big Government and Big Religion had nothing to do with its initial development and growth. It spontaneously evolved and became better, and better, and better. It matured into the fabric of modern culture with contributions from all types of individuals and organizations. Finally, it changed philosophical and economic reality by engineering technology that made the old way of thinking about and living in the world obsolete.

Notable Quotable

I'm still not very philosophical about open source. To me, its pretty pragmatic. [I have] a very strong belief that cooperation and open sharing of knowledge ends up resulting in better development. [...]

http://www.crn.com/

I guess you could call the belief in sharing of knowledge a philosophy, but I just think it's a fact. It's what differentiates science from alchemy or witchcraft. Anybody who doesn't believe in it is just wearing some serious blinders.

— Linus Torvalds

Debian IS the Design Science Revolution Incarnate

Debian is a Linux Distribution, that is, the packaging of the Linux Kernel, the GNU/Linux OS and lots of other software to make Linux easy to install, configure, and use.

Similarly, Debian has been an engineering initiative of spontaneous cooperation of individuals without significant assisstance from Big Business, Big Government or Big Religion. Now that Debian is leveraged by many derivative Linux Distributions it is changing the philosophical and economic reality through its engineering success.

Notable Quotable

Human beings, tiny though we are, are here for all the local-Universe information-harvesting and cosmic-principle-discovering, objective tool-inventing, and local-environment-controlling as local Universe problem-solvers in support of the integrity of eternally regenerative Universe.

— Buckminster Fuller

Notable Quotable

Today, all of human existence depends on the swift, world-around intercommunication system operating at 186,000 miles per second. We have transformed reality from Newton's "at rest" norm to an Einstein's 186,000-miles-per-second norm. Socioeconomically we have synchronized with the omni-intertransformative kinetics of the entire Universe.

— Buckminster Fuller

Dynamic, Organic, Upgradeable Software

To me, the priciples of Synergetics suggest that software systems like our ecology ought to be continually intertransforming and evolving to adapt to our ever-changing environment. That is, the traditional systems administration model of install it once and let it alone is a hold-over from a static Newtonian world view of "at rest". But building a dynamic model for software upgradeability is challenging given today's badly engineered software systems. So the concept of maintainability needs to be more broadly and diligently understood and applied.

Anticipatory Monitoring

Vital systems processes should be diligently monitored on an on-going basis to ensure correct functioning. Whenever possible, automatic feedback mechanisms should be employed to regenerate errant processes or automatically migrate them to servers (or virtual servers) where they work correctly. One element of this monitoring process includes reviewing log files for anomalies that might reveal subtle or nascent issues.

Cybernetic Control and Feedback Systems

- Continual review of software systems is essential to ensure on-going performance. As we all know, bugs are inevitable. Therefore, it is essential to continually audit software functions, their components, the monitoring infrastructure, failsafes including backups, and environmental factors such as human interfaces and use cases.
- Whenever possible, automatic cybernetic systems should be developed. But since this is a nascent art, it is likely that for the foreseeable future human intervention will be needed to catch the inevitable software deficiencies endemic to the immature state of modern software systems.

Note: **Cybernetics** is the interdisciplinary study of the structure of complex systems, especially communication processes, control mechanisms and feedback principles (Wikipedia).

Diligence and Meticulousness

"[Charles] Perrow is famous for his book *Normal Accidents: Living With High-Risk Technologies*, originally published in 1984. In it he argued that most major industrial disasters could be traced not to simple operator error but to the vulnerabilities of what he called complex, highly coupled systems, where each part depended on many others. He showed how small and apparently disconnected failures could cause such a system to fail catastrophically and unpredictably. So unpredictable are these systems that an effort to prevent one mode of failure may inadvertently create another one."

— Kenneth R. Foster from the January 2008 issue of *IEEE Spectrum*

Only diligence and meticulousness coupled with the vision provided by comprehensive thinking (vis-a-vis Synergetics) can help systems administrators prevent such failures

The Debian Policy Manual

This manual describes the policy requirements for the Debian GNU/Linux distribution. This includes the structure and contents of the Debian archive and several design issues of the operating system, as well as technical requirements that each package must satisfy to be included in the distribution.

http://www.debian.org/doc/debian-policy/

Debian Policy has the remarkable emergent property of addressing the issue of software maintainability better than any other effort of which I am aware.

Upgrades

- Since the upgrade process requires careful attention to illions of details to ensure correct operation, {pre,post}{inst,rm} scripts are necessary to address all cases.
- Major software upgrades such as from Apache 1.3 to 2.0, or PostgreSQL 8.1 to 8.3, or PHP4 to PHP5, special considerations are required. Ideally, both versions can live on the same system simultaneously until application software can be vetted against the newer version.
- Note: Debian GNU/Linux is the only operating system that I know of that has successfully managed to provide a “mostly” smooth upgrade path between major releases of the OS for over 10 years!

Upgrades

- Since the upgrade process requires careful attention to illions of details to ensure correct operation, {pre,post}{inst,rm} scripts are necessary to address all cases.
- Major software upgrades such as from Apache 1.3 to 2.0, or PostgreSQL 8.1 to 8.3, or PHP4 to PHP5, special considerations are required. Ideally, both versions can live on the same system simultaneously until application software can be vetted against the newer version.
- Note: Debian GNU/Linux is the only operating system that I know of that has successfully managed to provide a “mostly” smooth upgrade path between major releases of the OS for over 10 years!

Upgrades

- Since the upgrade process requires careful attention to illions of details to ensure correct operation, {pre,post}{inst,rm} scripts are necessary to address all cases.
- Major software upgrades such as from Apache 1.3 to 2.0, or PostgreSQL 8.1 to 8.3, or PHP4 to PHP5, special considerations are required. Ideally, both versions can live on the same system simultaneously until application software can be vetted against the newer version.
- Note: Debian GNU/Linux is the only operating system that I know of that has successfully managed to provide a “mostly” smooth upgrade path between major releases of the OS for over 10 years!

Debian Policy On Configuration

configuration file	A file that affects the operation of a program, or provides site- or host-specific information, or otherwise customizes the behavior of a program. Typically, configuration files are intended to be modified by the system administrator (if needed or desired) to conform to local policy or to provide more useful site-specific behavior.
conffile	A file listed in a package's conffiles file, and is treated specially by dpkg.

- Local changes must be preserved during a package upgrade
- Maintainer scripts can be used to manage configuration files, but then it must not be listed as a conffile and must not be part of the package distribution

Debian Policy On Configuration

configuration file A file that affects the operation of a program, or provides site- or host-specific information, or otherwise customizes the behavior of a program. Typically, configuration files are intended to be modified by the system administrator (if needed or desired) to conform to local policy or to provide more useful site-specific behavior.

conffile A file listed in a package's conffiles file, and is treated specially by dpkg.

- Local changes must be preserved during a package upgrade
- Maintainer scripts can be used to manage configuration files, but then it must not be listed as a conffile and must not be part of the package distribution

Debian Policy On Configuration

configuration file	A file that affects the operation of a program, or provides site- or host-specific information, or otherwise customizes the behavior of a program. Typically, configuration files are intended to be modified by the system administrator (if needed or desired) to conform to local policy or to provide more useful site-specific behavior.
conffile	A file listed in a package's conffiles file, and is treated specially by dpkg.

- Local changes must be preserved during a package upgrade
- Maintainer scripts can be used to manage configuration files, but then it must not be listed as a conffile and must not be part of the package distribution

Debian Policy On Configuration

configuration file A file that affects the operation of a program, or provides site- or host-specific information, or otherwise customizes the behavior of a program. Typically, configuration files are intended to be modified by the system administrator (if needed or desired) to conform to local policy or to provide more useful site-specific behavior.

conffile A file listed in a package's conffiles file, and is treated specially by dpkg.

- Local changes must be preserved during a package upgrade
- Maintainer scripts can be used to manage configuration files, but then it must not be listed as a conffile and must not be part of the package distribution

Sharing Configuration Files

- Packages which specify the same file as a conffile must be tagged as conflicting with each other. (This is an instance of the general rule about not sharing files. Note that neither alternatives nor diversions are likely to be appropriate in this case; in particular, dpkg does not handle diverted conffiles well.)
- The maintainer scripts must not alter a conffile of any package, including the one the scripts belong to.
- If it is desirable for two or more related packages to share a configuration file and for all of the related packages to be able to modify that configuration file, then the following should be done:
  - One of the related packages (the "owning" package) will manage the configuration file with maintainer scripts
  - The owning package should also provide a program that the other packages may use to modify the configuration file.

Sharing Configuration Files

- Packages which specify the same file as a conffile must be tagged as conflicting with each other. (This is an instance of the general rule about not sharing files. Note that neither alternatives nor diversions are likely to be appropriate in this case; in particular, dpkg does not handle diverted conffiles well.)
- The maintainer scripts must not alter a conffile of any package, including the one the scripts belong to.
- If it is desirable for two or more related packages to share a configuration file and for all of the related packages to be able to modify that configuration file, then the following should be done:
  - One of the related packages (the “owning” package) will manage the configuration file with maintainer scripts
  - The owning package should also provide a program that the other packages may use to modify the configuration file.

Sharing Configuration Files

- Packages which specify the same file as a conffile must be tagged as conflicting with each other. (This is an instance of the general rule about not sharing files. Note that neither alternatives nor diversions are likely to be appropriate in this case; in particular, dpkg does not handle diverted conffiles well.)
- The maintainer scripts must not alter a conffile of any package, including the one the scripts belong to.
- If it is desirable for two or more related packages to share a configuration file and for all of the related packages to be able to modify that configuration file, then the following should be done:
    - One of the related packages (the “owning” package) will manage the configuration file with maintainer scripts
    - The owning package should also provide a program that the other packages may use to modify the configuration file.

Sharing Configuration Files

- Packages which specify the same file as a conffile must be tagged as conflicting with each other. (This is an instance of the general rule about not sharing files. Note that neither alternatives nor diversions are likely to be appropriate in this case; in particular, dpkg does not handle diverted conffiles well.)
- The maintainer scripts must not alter a conffile of any package, including the one the scripts belong to.
- If it is desirable for two or more related packages to share a configuration file and for all of the related packages to be able to modify that configuration file, then the following should be done:
  - One of the related packages (the "owning" package) will manage the configuration file with maintainer scripts
  - The owning package should also provide a program that the other packages may use to modify the configuration file.

Sharing Configuration Files

- Packages which specify the same file as a conffile must be tagged as conflicting with each other. (This is an instance of the general rule about not sharing files. Note that neither alternatives nor diversions are likely to be appropriate in this case; in particular, dpkg does not handle diverted conffiles well.)
- The maintainer scripts must not alter a conffile of any package, including the one the scripts belong to.
- If it is desirable for two or more related packages to share a configuration file and for all of the related packages to be able to modify that configuration file, then the following should be done:
  - One of the related packages (the “owning” package) will manage the configuration file with maintainer scripts
  - The owning package should also provide a program that the other packages may use to modify the configuration file.

Hyper-Integration and Hyper-Configuration

- The *menu* system demonstrates important conceptuality for software integration and configuration.
- The `menu` package includes `update-menus` which provides services for both text-based and graphical menu building.
- End users can override (by adding, removing, or modifying) the sysadmins menus which can override each packages default menu.
- Each package containing a *menu manager* (a program that can display a menu) provides a script that can read menu files and compile them into an application specific menu.

Hyper-Integration and Hyper-Configuration

- The *menu* system demonstrates important conceptuality for software integration and configuration.
- The `menu` package includes `update-menus` which provides services for both text-based and graphical menu building.
- End users can override (by adding, removing, or modifying) the sysadmins menus which can override each packages default menu.
- Each package containing a *menu manager* (a program that can display a menu) provides a script that can read menu files and compile them into an application specific menu.

Hyper-Integration and Hyper-Configuration

- The *menu* system demonstrates important conceptuality for software integration and configuration.
- The `menu` package includes `update-menus` which provides services for both text-based and graphical menu building.
- End users can override (by adding, removing, or modifying) the sysadmins menus which can override each packages default menu.
- Each package containing a *menu manager* (a program that can display a menu) provides a script that can read menu files and compile them into an application specific menu.

Hyper-Integration and Hyper-Configuration

- The *menu* system demonstrates important conceptuality for software integration and configuration.
- The `menu` package includes `update-menus` which provides services for both text-based and graphical menu building.
- End users can override (by adding, removing, or modifying) the sysadmins menus which can override each packages default menu.
- Each package containing a *menu manager* (a program that can display a menu) provides a script that can read menu files and compile them into an application specific menu.

Shared Libraries

- Debian policy for binary run-time shared library packages (and their dependencies) ensures upgradeability during soname version transitions (`dpkg-shlibdeps`).
- Use `deborphan` to find old shared libraries to purge
- Many distributions botch maintainability due to an inadequate shared library policy. Red Hat has even botched libc! Many upstream vendors are not aware of how badly things can go for sysadmins if they don't bump the soname version every time the ABI (Application Binary Interface) changes. Study Debian's policy and infrastructure: they get it.
- TODO: automatic restarting of software when shared libraries are upgraded (especially for security reasons). But this issue is still not solved by any Linux distribution that I know of (`checkrestart` in the debian-goodies package helps).

Shared Libraries

- Debian policy for binary run-time shared library packages (and their dependencies) ensures upgradeability during soname version transitions (`dpkg-shlibdeps`).
- Use `deborphan` to find old shared libraries to purge
- Many distributions botch maintainability due to an inadequate shared library policy. Red Hat has even botched libc! Many upstream vendors are not aware of how badly things can go for sysadmins if they don't bump the soname version every time the ABI (Application Binary Interface) changes. Study Debian's policy and infrastructure: they get it.
- TODO: automatic restarting of software when shared libraries are upgraded (especially for security reasons). But this issue is still not solved by any Linux distribution that I know of (`checkrestart` in the debian-goodies package helps).

Shared Libraries

- Debian policy for binary run-time shared library packages (and their dependencies) ensures upgradeability during soname version transitions (`dpkg-shlibdeps`).
- Use `deborphan` to find old shared libraries to purge
- Many distributions botch maintainability due to an inadequate shared library policy. Red Hat has even botched libc! Many upstream vendors are not aware of how badly things can go for sysadmins if they don't bump the soname version every time the ABI (Application Binary Interface) changes. Study Debian's policy and infrastructure: they get it.
- TODO: automatic restarting of software when shared libraries are upgraded (especially for security reasons). But this issue is still not solved by any Linux distribution that I know of (`checkrestart` in the debian-goodies package helps).

Build Infrastructure

- Source packages should specify which binary packages they require to be installed or not to be installed in order to build correctly (except those tagged `build-essential`).
- The source code archive should only contain source code, never any files created during compilation.
- Do not include other packages that are also shipped separately inside your source archive, or if you do, please make sure that these can be reliably ignored. If a security issue is found in other included packages, it is far easier to rebuild one package than to scan the entire archive for all copies of this code and patch them individually
- See `http://wiki.debian.org/GettingPackaged` for more.

Build Infrastructure

- Source packages should specify which binary packages they require to be installed or not to be installed in order to build correctly (except those tagged `build-essential`).
- The source code archive should only contain source code, never any files created during compilation.
- Do not include other packages that are also shipped separately inside your source archive, or if you do, please make sure that these can be reliably ignored. If a security issue is found in other included packages, it is far easier to rebuild one package than to scan the entire archive for all copies of this code and patch them individually
- See `http://wiki.debian.org/GettingPackaged` for more.

Build Infrastructure

- Source packages should specify which binary packages they require to be installed or not to be installed in order to build correctly (except those tagged `build-essential`).
- The source code archive should only contain source code, never any files created during compilation.
- Do not include other packages that are also shipped separately inside your source archive, or if you do, please make sure that these can be reliably ignored. If a security issue is found in other included packages, it is far easier to rebuild one package than to scan the entire archive for all copies of this code and patch them individually
- See `http://wiki.debian.org/GettingPackaged` for more.

Build Infrastructure

- Source packages should specify which binary packages they require to be installed or not to be installed in order to build correctly (except those tagged `build-essential`).
- The source code archive should only contain source code, never any files created during compilation.
- Do not include other packages that are also shipped separately inside your source archive, or if you do, please make sure that these can be reliably ignored. If a security issue is found in other included packages, it is far easier to rebuild one package than to scan the entire archive for all copies of this code and patch them individually
- See `http://wiki.debian.org/GettingPackaged` for more.

Web Applications

- Debian's webapps-common policy is very good, but few applications support it. Which means there are too few web applications effectively included in Debian. See `http://webapps-common.alioth.debian.org/draft/html/`.
- Webapps should follow the FHS (Filesystem Hierarchy Standard): configuration files in /etc, content (dynamic & static) in /usr/share/PACKAGE/www, application specific files in /usr/share/PACKAGE, etc.
- Web packages should support registration with the system administrator's chosen httpd server under multiple IP based or name based virtual hosts.

Web Applications

- Debian's webapps-common policy is very good, but few applications support it. Which means there are too few web applications effectively included in Debian. See `http://webapps-common.alioth.debian.org/draft/html/`.
- Webapps should follow the FHS (Filesystem Hierarchy Standard): configuration files in /etc, content (dynamic & static) in /usr/share/PACKAGE/www, application specific files in /usr/share/PACKAGE, etc.
- Web packages should support registration with the system administrator's chosen httpd server under multiple IP based or name based virtual hosts.

Web Applications

- Debian's webapps-common policy is very good, but few applications support it. Which means there are too few web applications effectively included in Debian. See `http://webapps-common.alioth.debian.org/draft/html/`.
- Webapps should follow the FHS (Filesystem Hierarchy Standard): configuration files in /etc, content (dynamic & static) in /usr/share/PACKAGE/www, application specific files in /usr/share/PACKAGE, etc.
- Web packages should support registration with the system administrator's chosen httpd server under multiple IP based or name based virtual hosts.

The Challenge of Comprehensive Anticipatory Systems Administration

- Software maintainance is still an unsolved problem in systems administration.
- Software developers need to understand the key issues so they can provide a better infrastructure for maintainability.
- The type of Comprehensive Anticipatory Systems Administration inspired by study of Buckminster Fuller's *Synergetics* and the *Comprehensive Anticipatory Design Science Revolution* provides a roadmap to developing eternally regenerative software administration capabilities.

The Challenge of Comprehensive Anticipatory Systems Administration

- Software maintainance is still an unsolved problem in systems administration.
- Software developers need to understand the key issues so they can provide a better infrastructure for maintainability.
- The type of Comprehensive Anticipatory Systems Administration inspired by study of Buckminster Fuller's *Synergetics* and the *Comprehensive Anticipatory Design Science Revolution* provides a roadmap to developing eternally regenerative software administration capabilities.

The Challenge of Comprehensive Anticipatory Systems Administration

- Software maintainance is still an unsolved problem in systems administration.
- Software developers need to understand the key issues so they can provide a better infrastructure for maintainability.
- The type of Comprehensive Anticipatory Systems Administration inspired by study of Buckminster Fuller's *Synergetics* and the *Comprehensive Anticipatory Design Science Revolution* provides a roadmap to developing eternally regenerative software administration capabilities.

The Challenge of Comprehensive Anticipatory Systems Administration

- Software maintainance is still an unsolved problem in systems administration.
- Software developers need to understand the key issues so they can provide a better infrastructure for maintainability.
- The type of Comprehensive Anticipatory Systems Administration inspired by study of Buckminster Fuller's *Synergetics* and the *Comprehensive Anticipatory Design Science Revolution* provides a roadmap to developing eternally regenerative software administration capabilities.

Integrity

Integrity is the most continually important issue in human affairs. We must, each of us, persevere to consider and re-consider our values and the value of our actions. Integrity implies honesty with oneself and honesty with others. Integrity implies truth: being true to the facts of experience and true to oneself and true to Humanity and true to Earth and true to Universe and true to God. Integrity requires consideration and re-consideration of the whole and all of the parts and their synergetic interrelationships. Total, complete integrity is impossible for finite humans to fully achieve. For human beings, Integrity is a process that requires perseverance, discipline, dedication, and continual re-dedication.

What you can do to help

- There is a lot of work to do. Since Debian is the world's leading institution working to solve these issues, I recommend becomming a Debian developer and working to provide more maintainable software to the FOSS community.
- In whatever project you work on, look deeper into the complex of maintainability issues that threaten system stability.

What you can do to help

- There is a lot of work to do. Since Debian is the world's leading institution working to solve these issues, I recommend becomming a Debian developer and working to provide more maintainable software to the FOSS community.
- In whatever project you work on, look deeper into the complex of maintainability issues that threaten system stability.

Notable Quotable

Initiative-Taking

The things to do are: the things that need doing: that you see need to be done, and that no one else seems to see need to be done. Then you will conceive your own way of doing that which needs to be done – that no one else has told you to do or how to do it. This will bring out the real you that often gets buried inside a character that has acquired a superficial array of behaviors induced or imposed by others on the individual.

— Buckminster Fuller

Thank You

Thank You!

Any Questions?

http://www.CJFearnley.com/comprehensive.anticipatory.SA.pdf