Script for “agile sysadmin” talk. Go to the next slide for each paragraph. Apart from this one.

Hello, apologies for the scripted talk, but time is too tight for me to ad lib. This is not a straightforward technical talk, more of an open question. I'd like to talk to you about an idea I've been mulling over. We could call it Agile System administration...

...or perhaps eXtreme System Administration...

...or Lean...

...Lite?...

...next generation...

...the name's not important.

System Administration today is full of challenges, which as always are about increasing stability, performance and so on, while keeping costs under control.

There are new disruptive technologies which bring new complexity, mostly because they increase the number of “moving parts” to manage

and there are the age-old issues of trying to keep a whole stack of infrastructure and applications working together while staying on supported versions. I'm sure we've all come across systems where you have to coordinate versions of webserver, database, vendor code, and operating system on multiple servers, to keep the complete stack compatible and supported.

So how to get better at this? How do you measure yourself against best practice? Or just cope at all?

Well, it's interesting how much thought goes into improving the software development process.

There are many people researching and publishing about better programming, “thought leadership” for if you like.

In recent years, there have been many innovations in development techniques...

...many of which are associated with the Agile methodologies. On top of those techniques, the Agile Manifesto emphasises flexibility and cooperation over formal plans and contracts.

I don't think the concepts from Agile programming map exactly onto Sys Admin tasks

For example, Test Driven Development says that everything should have a test ... but if you've just been shipped the latest version of your database, you can't test every aspect of its functionality and performance – that's what you pay the vendor for. You should of course test what you can, but just installing a test environment can be hard work. Can we get closer to automated testing?

What does map is the mantra never to ignore a broken test – in Agile development, if your new code won't pass, fixing it is the first priority. But how many times have you seen regular jobs and environment health checks that give warnings which everyone “just knows to ignore”?

“value working code over documentation” - probably a rather

bad thing in the support team!

pair programming – I don't think we'll have pairs of admins making every change, or two people to answer every support call...

and so on and so on – these ideas from development make you think, but mostly they don't really apply directly.

So by contrast to all this fevered thought applied to development, what do we Admins get?

More acronyms than you can shake a stick at! These are often quite good things, but they tend to be high-level, aimed at management, and not into the technical detail. Frankly, they are a complete turn-off for most techies.

So what else? There are many many books on how to improve development, and that's excluding technology-specific books like “How to program Java”. There are very few equivalents for Admin. Obviously if you have “How to administer Linux”, you can glean some ideas, but it will tend to focus on how-tos for specific tasks. The main way is

learning from your peers – come to an event like this and chat.

Just to reiterate, what I'm looking for is something between ITIL “you should have a CMDB” and specifics like “how to maintain a service in the Solaris SMF”. Developers can get very detailed advice on how their code should look, how big a class should be, code “smells”, etc – they can choose to disagree, but there's a starting point.

So where's the thought leadership for Sys Admins? Earlier I showed a list of well-known development authors, which could have been two or three times longer – could you name a couple of equally well-known leaders for System Administration?

Maybe development is more susceptible to analysis, or maybe the people who do it are more interested in theories.

But surely there must be something to learn?

I want to improve: what are the best and worst practices, what are the metrics to worry about?

So here are some ideas for specific practices. I could probably talk for an hour about each one, but to illustrate the sort of thing:

Configuration Management – ITIL talks about CMDBs, but let's get hardline, let's insist that the CMDB controls the environment – don't go scanning the network for what's out there, if it's not in the CMDB it doesn't run, full stop.

Charging: You get a phone bill from BT or Vodaphone, it can list every call or text and how much it cost. We tend to tell users “you're paying X million pounds for your databases”, including machine costs, licenses, and support ... and then when you ask your users to clean up old data, or ask your developers to move off the legacy version, they are unwilling because they are too busy, i.e. there's nothing in it for them. So automate charging for each MB of storage, for each server or CPU, and charge more for legacy systems if they cost more – so if the business prefers to pay for the legacy rather than an upgrade, that's their call.

Monitoring – this is a wheel that's always getting reinvented –

what to monitor and how. This is our equivalent of Test-Driven Development – so be radical. Insist on in-depth monitoring of every component. No false positives or negatives should be tolerated.

Application deployment – surely this could be better understood – at present it is usually a case of doing what your tool enables.

Documentation – here's one that people like to argue about – what's your ideal? Most places I've seen have 3 or 4 separate legacy systems, none of which do the job.
[Pause] So these are some of my ideas – disagree with them if you like, but we could at least start the debate. What's the best practice? How do you measure your success?

People like metrics, particularly managers. Here are some traditional ones, but they are at the highest level, sort of external to the systems.

I don't know if we could get better ones – track how comprehensive your monitoring coverage is, or number of jobs failing, perhaps? But I don't know – just as counting

lines of code is more or less discredited, perhaps these things are just distractions...

And some tools would be nice.

Sys Admin tools don't seem to have the excellent breadth of coverage you find in development. Perhaps that's for the obvious reason that developers tend to develop stuff. And the quality of the proprietary tools is often terrible, though I suspect that's partly because companies are irrationally keen to customise apps like ticketing systems, and don't fix the problems later.

So some suggestions:
A CMDB that works for the hands-on administrator who is making changes, but also allows detailed reporting on the installed environment.
A task manager – how many jobs do you have running under cron that just email if all goes well, so you've got a stack of mails to check on a Monday morning?
Ticketing – some of the worst human interfaces I've ever seen have been on ticketing systems.
Documentation – make it easy to enter docs and easy to find

them – can it be all that difficult?
...and so on!

So I'm done. What do you think? Am I missing something obvious, am I just a hopeless idealist, or is there something in this? The link there is brand new – I added that yesterday because I suppose if I want to stimulate debate, I need to make a forum for it ... I'll post these slides there and I guess my Christmas task is to flesh out some more about these ideas. Comments will be welcome!

(Note: there are a couple more slides that follow that I cut out of the talk. A summary of the Agile Manifesto – go look it up if you want to know more. Also an attempt to illustrate a "compatibility stack", i.e. the pile of components that have to work together for a particular product I work on – Documentum, EMC's document management product. This is no ill reflection on that product, just the nature of a modern complex system with many well-separated pieces of functionality. EMC is reasonably good about supporting a range of versions, though never as good as customers would like; that's commercial reality.)