12 Steps to a Better Virtualised Desktop Service

andrew.wood@gilwood-cs.co.uk

August, 2012

Version 3.0c

Summary

A virtual desktop is a common goal. The functional requirement of that goal – an experience as good as or better than a physical desktop for both the user and the administrator

How do you achieve that?

In a growing environment of "consumerisation of IT" how do you ensure that the delivery of your virtual desktop service is easy to use, effective, and reliable?

How does, how will, your virtual desktop environment score as you take 12 steps to a better virtualised desktop service?

Foundation

A virtualised desktop environment offers a number of benefits: and an often cited one is “better management”. Virtualised desktops allow for centralisation of services and better automation. Happy Days. However, many administrators find that, far from being easier, a virtualised desktop environment – be it based on hosted desktops (VDI) or session based desktops Microsoft Remote Desktop Session Host (RDSH), the desktop virtualisation technology formerly known as Terminal Services – can be as complex, if not more complex, than what they had before. Bad Times.

In August 2000, Joel Spolsky wrote an article that helped to evaluate the quality of a software team. Joel is a software engineer and writer; he is the author of *Joel on Software*, a blog on software development. Joel worked at Microsoft as a Program Manager on the Microsoft Excel team, where he designed Excel Basic and drove Microsoft's Visual Basic for Applications strategy.

A simple thing about Joel's test was that it took a short time to do, and while he accepted it wasn't foolproof, the concept was if you got the twelve things right, you’d have a programming team that would more than often deliver.

Your *foundation* was right.

A virtualised desktop service often gets a bad press. It is not unusual to suffer from project stall due to user experience issues. What appeared to be a great pilot can flounder once full roll out begins. Or, initial roll-out may appear fine, but 2-3 months down the line there are general grumblings of users about “slowness” or “unreliability”.

As the current market leader, Citrix (with its XenApp and XenDesktop products) often finds itself as the default root cause for a virtualised desktop problem. It is not unusual for a virtualised desktop issue to be 'Citrix’s' fault: when more thorough investigation shows otherwise.

Likewise, it is not unusual to find prospective virtual desktop administrators asking how they install their environment so that it will run 'just fine'. Questions are asked about "what are the list of hotfixes" needed to 'make the farm stable'; “how can I see if the servers are performing well”, “how can I work out why they are performing poorly?”

There is not a *single* thing that can help solve all issues – so maybe it would be better to look at a test for factors in a successful farm.

What are the foundations of a good virtualised desktop environment?

So here is Joel's test - as applied to virtualised desktop environment. As with the original test, the higher the score the more likely you have a disciplined environment, a good team and a stable service that consistently delivers. A low score and you need to take a look at the team and operating environment; your farm - even if it’s only a couple of servers - could be causing you more work than it should and your users more grief than even they deserve.

Contents

1. Do you use Source Control?

'I'm not a programmer - this is a stupid test' may be your response to this first question. You may be asking why you need a method of keeping track of versions of batch scripts, GPO templates, installer package scripts, login files, program launch scripts, application compatibility scripts and indeed, build images themselves.

Well for a start, that's a lot of **stuff**. How do you track what the changes are between versions of say, your login scripts e.g. the version that you had last week that worked, and the updated version you have now which doesn't? Before you start editing a login script, do you take a backup? Do the other guys on your team?

Source control can be a bought product; it can be a free product. You could use something like CVS, Git or Mercurial. I've also used frameworks like Microsoft SharePoint or Joomla and associated document/version control systems.

The goal is to have a repository of all your farm components. Scripts have a version and a state of 'being in use/being in development'. Within the script, at the start, there is a note on what was changed, why, when and by whom. Documents, designs, test plans – all the same.

Same goes for build images. You will likely have a library of images by version number – especially in a hosted desktop environment. This is fine - but also, where are the supporting notes on what has changed between those versions and why?

This way you track changes, of who did what, when. If need be, you can get back to what was working in the event of disaster but ultimately you have a record of changes to assist in troubleshooting and keeping everyone on track. Virtualised images do allow for rapid roll back – but not all components of your farm are virtualised images, and within those images there are inherent changes and updates that need to be logged as an issue may not always surface in the next release, but sometimes in releases three or four stages down the line.

2. Do you Automate your Build Process?

People get stuff wrong - this is a cast iron FATC[1]. People cut corners. Give someone a list of steps on bits of paper and a pile of CDs and something will be missed somewhere. And you will lose a CD. The process of getting a server/virtual machine from raw disk/empty vhd to working operating environment should be as few steps as possible – as this means there are as few errors in the core environment as possible.

[11] Not me tho' – FATC was technically a humorous example – but mistakes are indeed easily done

OS/application virtualization is not full automation; it is delivering a prepared package. It makes it easier to deliver, it isn't automating. Someone has created the package/sequence that has to be automated.

A huge benefit of automation is that every step in the process is done in the same chronological order. This in turn leads to a more stable and manageable environment.

OS Automation

There is a slowly developing trend to deliver pre-configured virtual images. While this is incredibly useful, there are still many instances where an environment needs to be configured from scratch each time every time. Moreover, how does that pre-configured virtual instance come about in the first place?

Citrix's Provisioning Services, for example, allows you to create an image to be automatically deployed to physical or virtual machines over the network. Citrix Provisioning Services is an extremely valuable tool in rapidly deploying an image to your server estate: but even this service relies on an image being created and available, stable and tested prior to be it being delivered.

Citrix have worked hard to streamline their application installations. Installation and configuration are typically separate tasks. This task division provides flexibility when using provisioning tools and disk imaging.

But the applications (or agents) need to be installed into an OS. For the Microsoft OS you can use Microsoft's own tools such as the Windows Automated Installation Kit or Windows Deployment Services. Or tools from vendors such as Visionapp, RES Software (Automation Manager) and Symantec (Altiris Client Management Suite) can be utilised too.

The goal should be to automate the process of taking raw tin to useable environment to make it reliable and reproducible across environments – be they virtual or physical.

Application Automation

Your build is not just your core operating system. How do you install your applications? Is it completed by hand? Is it scripted? What is the process of users requesting a new application?

A great technology to improve and automate the delivery of applications to users is application virtualisation. Application virtualization is an umbrella term that describes software technologies that improve portability, manageability and compatibility of applications by encapsulating them from the underlying operating system on which they are executed. A fully virtualized application is not installed in the traditional sense, although it is still executed as if it.

Do you use one of the application virtualisation tools – such as Microsoft's App-V, Citrix Application Streaming, Novell's ZenWorks, Spoon Virtual Application Studio or VMWare's ThinApp? Yes, there is a process required to transform application installs to be virtualised – but there are also tools to automate and manage that. VMware's ThinApp Factory for example, is a virtual appliance that brings centralized administration and automation to the process of creating virtualized Windows applications with when you use ThinApp. Products such as Citrix's AppDNA or Quest's Changebase can assist in reducing the time to convert applications to a virtualised environment. Typically, these solutions require that the IT team prepare an application prior to it being available. There are vendors who are offering solutions that allow you to let users install applications themselves in a more managed and controlled fashion. For example there is AppSense's StrataApps, Liquidware Lab's Flexapp or Citrix's Personal vDisk.

That is not to say that application automation can only be achieved through application virtualisation. The goal is to have an environment where requests from users to have applications are automated as much as possible so that the delivery of their application is quick and reliable.

3. Do you Rebuild or Patch?

Do you apply an operating system hotfix to a working environment - or do you restart and rebuild the environment from scratch?

If you apply patches to a 'live' environment you can often encounter unexpected problems. Files can be left open or locked and so don't get updated correctly. In 'live' environments you also may have different hotfixes modifying the same file, so the order of the hotfix installation is important. Is that hotfix order applied to new servers - or do new servers have the hotfixes applied in a different order?

Applying patch after patch can therefore mean that your operating environment is not as clean and stable as you originally intended/designed.

Ideally, incorporate the hotfixes into your automated build process and rebuild the environment – so it is a fresh installation each time. With automated tools this 'rebuild' time (i.e. a working, tested environment deployed to servers) can be much less than you would expect.

Now, you have a reliably clean environment to apply the patch to, you can better manage the order of hotfixes. The server itself is in a better state to be called 'reliable' and you can be more confident that OS A has been configured as OS B.

4. Do you have an internal bug & resolution database?

Keep a track of issues on your system – both to make sure nothing gets forgotten and to enable you to best prioritise fixes. You can of course extend a simple bug tracking database to include other information – such as configuration settings, or 'how to' articles – building your own knowledge base.

Within any virtual desktop environment it's more likely you will be collecting application issues. It is a character building lesson for many to find out that there are many piece of commercial software that have not always been thoroughly tested and do not work faultlessly. So, as well as keeping a check on your own scripts, your own OS build environment it will assist to you to record issues with the whole environment – including applications.

Ultimately, do you have something to help you not have to reinvent the wheel? To help track requests for support from third parties?

At a base level your issue log should record:

- Steps to reproduce the issue
- What you expect the behaviour to be (alright it *shouldn't* do **that** but... what should it do?)
- Observed behaviour.
- Who is dealing with the issue?
- Is it fixed?

Microsoft SharePoint has a resolution task list option – that could be a start if you've nothing but email coming in. Or, you can manage the process better - third party helpdesk/bug tracking systems are available to suit various budget sizes, and there are even freeware support tools such as Bugzilla and Spiceworks

As your issue database grows it should be the first port of call when looking at an issue – has it already been reported? Has it been resolved before? Yes, the Internet and forums are a great resource, but have you checked internally before going out to the wider web? Often, an internal repository not only allows quicker problem resolution, it can act as a training aid or the start of a justification for additional resources.

Should you have the appetite for it, this approach can also take you on the glorious road to ITIL Incident and Problem Management practices and all the joy that brings. ITIL advocates that IT services must be aligned to the needs of the business and underpin the core business processes. It provides guidance to organizations on how to use IT as a tool to facilitate business change, transformation and growth. You can find out more about ITIL at the Official Site

5. Do you deploy hotfixes only to fix a given problem?

A hotfix – be it a Citrix, Microsoft, VMware or an application hotfix is a set of files that have been changed in some way to resolve a specific issue.

There is a belief that applying all the latest patches will somehow make an environment more stable as it will be more 'up-to-date'. But, a hotfix has been developed to fix a specific issue or provide a specific piece of functionality. It will rarely have been stress tested; it's unlikely to have been exhaustively examined on a wide variety of systems.

Think about the hotfixes you are applying. If it's not relevant to your environment skip the hotfix. It's one less thing for you to test before its deployment; it's one less possibility when you are trying to troubleshoot a particular problem.

6. Do you have a schedule for new application/fixes deployment?

Schedules are indeed A Good Thing. New software comes out, patches become available, changes are requested that need to be installed.

Often virtualised desktop administrators complain of having to 'fire-fight' rather than 'develop' their farm. Or, that the farm appears unplanned or haphazard and that when they have some time they are going to 'sort it out'.

This is nearly always because they've been asked to update or change the farm, but didn't have a schedule. You may say 'I'd never get away with that – people want apps delivered ... yesterday'. But without a decent schedule it's very unlikely that the application will be delivered tomorrow. And of course you know when tomorrow comes?[2]

A basic schedule could simply outline the steps and time needed to test the application install, validate the installation, and incorporate into the existing build before rollout to the other servers.

For any application:

- Who is the business' application owner? Who is responsible for requesting the application? Given your organisation's application portfolio – why is this new one different? Responsibility for an app isn't an IT problem – it's a business problem.
- Who engages with the software supplier – is it IT, is it someone else?
- Who will test it? More on this later.

In that basic schedule you can identify who will be needed, when and for how long. What this means to the business is that the application is delivered to users as expected. It's more likely to be reliable, it's less likely to have users complaining because the app doesn't work – or breaks some other application. Certified change management people might be nodding their heads sagely: it's not always about getting certified and formal procedures, sometimes it's simply about understanding a workflow to be able to allow everyone to get home in the evening and enjoy a weekend.

[2] ..the answer is never. I'd not accept "never dies", but might roll with "tomorrow" if you were being sassy

Such a process can be extended to create a workflow that puts all the business requests for change to a committee, a ultimate change authority board that analyze the risk, schedule updates and discuss the rollback scenario. Such a group shouldn't just consist of IT people – this should involve the business. How formal this team needs to be depends on your company size. Rule of thumb – if the process isn't working informally, formalize it.

7. Do you have a documented build specification?

'..but it's an automated build …you made me automate it in test 2 - I spent weeks doing it, why do I need to document it as well?' you may ask. Like Joel says, "*creating a specification is like flossing – everybody agrees that it's a good thing, but nobody does it*".

Tough ;) A specification allows you to sanity check proposed changes before you implement them. It also means that someone else can validate or review the changes made at a later date. In the event of a real disaster – you could even go back to the documentation to help start from scratch.

More importantly, without a specification how does the business (you, the management team, the finance people, the users, maybe even Doris who makes the tea) know what is to be delivered, and if it's the right thing?

If reams of paper chill you to your bone, be practical. Documentation need not be a voluminous masterpiece explicitly referencing every single script and interaction. 'Documentation' at a code level could include good comments, explaining what a function does, and how; documentation at a design level showing how the components of your environment interact; documentation from a business perspective stating what the service is expected to do.

Document configuration settings if they can't be exported to an html file. There are some tools to help – Carl Webster has a number of PowerShell scripts that can help document a Citrix environment (e.g. PVS, XenApp) , or XTS's Introspect. Alan Renouf has also produced some great reporting scripts for VMware. Feel free to chime in with others.

Regardless, it's a good idea to enforce the idea of 'no changes implemented without the documentation updated first'. It is not unusual for the process of reviewing/updating the documentation highlighting the fact that the change will impact something else.

And of course, all build specs in the source control environment - but you already knew that.

8. Do you have a helpdesk policy?

You can work on a problem best if you are uninterrupted. Do you have a single email address to take all requests for changes and requests for support? Even if you're a small team do you have a dedicated phone available for support calls? Do you only schedule looking at a problem once it's entered into the helpdesk? Do you get your helpdesk (if you have one) to ask the useful questions beyond resetting the national grid interface? Do you have issue priorities? Do you enforce them?

It may be easier for users to call you directly, or stand at your desk looking mournful until you click the 'any-key' – but if they are interrupting you they are preventing someone else from getting their issue resolved, they are potentially delaying not only you, but maybe even a whole department.

Does the helpdesk know what questions to ask? Ideally by the time you've got the call in front of you, the call details mean that you've got an understanding of the situation – this way you don't spend a lot of time getting the information from a now irate user (because they've had to go through this twice). Working with the helpdesk to properly identify the issues means that you and the user are making best use of your time and the company's money.

A helpdesk policy and informed call takers lets the business know that issues will be dealt with in an orderly and therefore timely (and hopefully cheaper) manner.

9. Do you use the Best Tools money can buy?

"But tools are expensive... My boss won't let me spend any money."

Look at your issue log. Work out how long you have spent on particular problem and review how much time it takes to resolve issues. By all means discover your workaround, but is there a utility that could be bought in to do the work to save you time and effort and to help the business to get a faster resolution?

For example:

- **Profiles and User Configuration:** A large number of profile issues or printing issues can be fixed with third party tools. AppSense, Liquidware Labs, RES Software, Tricerat, VirtualInfo and others all have ready-made solutions again, to suit various pockets.

- **Printing:** Despite this being "the future"; and despite us having the facility to wander around using a tablet to ~~view the star ship's schematics and battle readiness~~ read documents; printing is still a big part of enterprise. Printing in a virtual desktop environment can be challenging – but there are solutions from

Citrix, Quest, UniPrint, Thinprint, and Tricerat that can assist in ensuring that the user experience is good, and the administrative overhead is reduced.

- **Monitoring:** more on this later – but do not, not monitor...

Schedule in development time. This will be easier now you've a helpdesk policy and improved automation. This could be a step in its own right – but I'm going to stick to 12. Make it a goal to introduce appropriate tools to help resolve issues quickly, or even help you remove issues altogether.

Microsoft have a number of Best Practice Analyser tools that can be applied to environments to help give you an indication of where tweaks can be performed.

Happier working users are more productive than sad, broken users. Yes, there is a cost to the business for introducing such tools, but there is also a cost to having users not being productive. If you were to consider how much time it takes over a period to fix issues, to have users sat with unresolved issues and translate that to a cost figure, it's not unusual for the business case to be simply 'we will save money' by spending X.

10. Do you have a dedicated test environment and users willing to test with test scripts?

A test environment is core to making sure that application, update, or changes you are about to unleash on your users will not come straight back at you. Your test environment is there so that you, your team and the business have the opportunity to know that what you've scheduled to do is going to work as expected.

Test 'scripts' need not be scripted, they could be documented procedures. The document outlines the steps to perform the test, and potentially the team/people/manager responsible for the application and its testing.

Automated scripting (such as that with AutoIT. Citrix Edgesight for Load Testing, Denamik's Loadgen and Login VSI's Load Testing) allow you to run through tests and save them so that you can run them more conveniently and ideally with multi-user loads.

Citrix XenServer, VMware ESXi, Microsoft Hyper-V – server virtualisation technologies mean that hardware resources don't need to be excessive in order to maintain these environments. And you obviously benefit from getting a better understanding of these technologies if you've not had a chance already.

Automated scripting gives you scale – but the best way to test your deployments is to give it to expert users. For any application (as with purchasing/introducing that application):

- Who is the business' application owner? Who is responsible for the application? Responsibility for an app isn't an IT problem – it's a business problem.

- Who is responsible for contacting the software supplier if you find an issue?
- Who is able to test and confirm that the application works? Who in the business is the key user?

The goal of your test environment should be to reduce the impact of a new change request to users, and help make sure that they are more – or at least not less - productive. Don't exclude the business from this process; get your users involved because they are the best people to test the application.

11. Do you give prospective administrators real world scenarios to figure in an interview?

Question and query people who are going to be working on your farm. Don't simply print out a set of test questions and expect verbatim replies. You want problem solvers, not parrots.

Ask them not only what they've done – talk to them to gain evidence; ask them to resolve, or at least provide some insight, into issues from your helpdesk; or prepare a brief design for the specification of the service that you want. If they've stated they have qualifications - where are they?

Lots of people, your whole business in fact, depend on your server farm operating efficiently. Make sure the people feeding and watering it understand how it works, and how to go about working effectively should it go wrong.

Joel has a Guerrilla Guide to interviewing that you may find of use, or not – but I think it's a useful read. Yes ok, it's overtly about programmers. But consider the sentiment – "if the team doesn't get along, we'll never be able to work together. "*Most importantly – when you hire in principle, it's simple. You're looking for people who are*:"

- Smart, and
- Get Things Done.

Now, that doesn't mean (always) qualified. I'd like the certification fees to come down – it's overly expensive to be qualified. I'd like an easier method for qualification verification. For me most tests don't focus on being smart and getting things done but rather on reiteration of facts designed for service providers rather than implementers - but the testing medium is what it is. By all means, rage against the machine.

12. Do you monitor your farm?

How do you know what your users think about using the service? How do you know that the new memory in the servers has made a difference? You've been told that the end user performance of your virtualised desktops/apps is 'poor' – but what performance? Is it for all users, or specific users? Is it the connection? Is it the backend applications? Is it specifically rubbish for some people and fine for others? Is it an application you have control over? Is it a consistent issue, is it variable?

There are essentially two types of performance. Computational performance (resource use like CPU, memory, disk, etc.) – which pretty much everyone does. There is also perceived performance (i.e. response times): which many don't. The key for a successful virtualized desktop environment is to measure both. So. monitoring is more than simply checking on the memory and CPU of servers/hosts. This is important for sure, but you also need to consider at a base level:

- Application performance
- The user experience

Application Performance

You should have a monitoring service reviewing the performance of your virtual desktop environment servers that reports back on performance and usage. This may be as simple as Performance Monitor running on the servers showing CPU, memory, disk and license count usage Or if you're a Microsoft RDSH admin you can utilise tools specifically designed for their environment with utilities like Terminal Services Log Or, you may decide to invest in products such as Citrix's EdgeSight Lakeside Software's Systrack, Liquidware Labs' Stratusphere that give you an indication of the performance and reliability of your virtualised environment. That said, a virtualised desktop service rarely runs in splendid isolation – consider wider monitoring tools such as those from egInnovations, or Splunk to consolidate information from all components in your environment to provide more holistic information

By collating statistics and monitoring those stats regularly you can confirm performance issues with the environment, or spot trends in increased demands to better plan to introduce updated services or isolate troublesome updates or applications.

Or wander round the office with a big smile because everything is just dandy.

User Experience

Understanding User experience is key when delivering virtual desktop environments. Yes, many of the performance monitoring tools will provide metrics on user experience – but the best people to ask are the users themselves.

Ideally you also have the facility to gain user feedback (this might be a survey in Sharepoint, or services such as SurveyMonkey). This way you get a more direct understanding of how users view the service - with the business benefiting in the fact that there is a direct feedback from the end user to services.

The above will tell you the performance of your farm, but I'm going to add a third – I also think you need to consider the cost of delivering your farm. In which case there is an additional item to monitor – environment and application usage.

Environment and Application Usage

A common issue when moving to a virtualised desktop environment is licensing of the environment – which Microsoft OS client licenses are needed? Two useful resources here are

- Microsoft Licensing for VDI Environments
- Microsoft Licensing for Virtual Environments

As importantly, is the licensing of applications still valid, or has this changed? Indeed – what is your application usage? How often is an application used, when it is used, and by whom?

To understand this – monitor which applications are used and when. Not only OS based apps, but web based applications too. Why? Because where is data being stored outside of your network, who is using it and how often is it being used aren't just questions that need to be answered for internal applications.

Four ways to use the test

1. Rate your own farm, tell me how it rates and how you think your farm is performing – then I'll know if it's a good test and I can tweak it if need be with of course, credit going to you. How? Take each item and score yourself 0-5 points where :

   - 0 is "nope, don't do this at all" and,
   - 5 is "really happy with what it is that we've got here – in fact, we're mint[3] at it".

   How do you fare?

     - <30 and you're maybe entitled to take a half day thinking about where it could all be better.
     - 30-45 is the ball park although you've likely ideas for improvement;
     - 45+. Well done you. By all means share your processes, setup and insights as (by this test) you're doing very well.

2. If you're going for a role managing/implementing an environment, ask the prospective employer the question above. If they've a really low score, make sure you'll be able to implement changes – otherwise it could well be frustrating.

3. If you're about to do due diligence, or review on an environment hopefully this test will act as good rule of thumb.

Finally, by all means feel free to comment on the test. But please don't say things like 'ooo but you must have printing' as I think it should be a generic test – not all environments have or need printing (yes I know there was a bit about Printing in "tools" but that was an example of tools..).. ahhhh, I know you get it really.. and you're just messing.

Enjoy.

a.

[3] Mint = most excellent

Appendix: Version History

3.0 July 2012 Include wider range of virtualised desktop delivery models. Change the Pubform reference as it has been changed to E2E...finally. Thanks to @ChrisJMarks and @IngmarVerheij for their time, effort and additional insight in reviewing the document.This release was mostly completed while listening to Bruce Springsteen.

2.5 Feb 2010 We don’t talk about this release.

2.0 June 2009 Updated sections on tools and documentation, made other nonsensical bits less nonsensical

1.1 Feb 2007 Updated sections on tools and documentation, made some nonsensical bits less nonsensical; thanks to paul@watson-online.net

1.0 Jan 2007 Initial Release – thanks to yurihaak@hotmail.com; mailto:theartvark@gmail.com; graham@galdek.co.uk for their time, effort and additional insight in reviewing the document