Application Intelligence: A Look at AppTitude from App-DNA

Abstract

Let us be clear up front. There is no magic bullet to eliminate the pain of application migration. *AppTitude* from App-DNA is a unique and valuable tool for acquiring and managing "*Application Intelligence*" on your software applications, with a strong emphasis on understanding compatibility and conflict. It analyzes application installers against a long list of criteria and databases the results for reporting. This is the kind of tool to acquire prior to the start of any kind of large migration of applications. It will aid you in sizing the work involved. More importantly, it will help you to understand, early on, which applications will require significant work by your premium talent, which ones "should" go smoothly, and which ones to handle via "Plan B". Because you can add your own experiences to the database beyond what AppTitude detects, this tool grows in value as you work your way to that next migration that typically hits you just as you finish your current one.

The value of the product is in the reduction of time and risk for migration projects. Pricing for AppTitude is based on the number of applications you analyze and the number of modules you need. A module might be checking against Vista, or against App-V. Although the product is often sold into Enterprises, they also have an active partnership with consultants who purchase it to use as part of their practice – providing the results to customers as a service. The reports generated are professional enough for a consultant to deliver to their customer as part of a paid-for analysis service. I believe that this product may also be invaluable to ISVs to test their own product to help ensure the software is being written and deployed following Microsoft best practices and procedures.

In this article, Tim Mangan (who happens to know a lot about application issues) takes App-DNA for a spin against a suite of applications and reports on what he found.

Introduction

Software applications are like an "old school" boss. They might have been top notch in their day, but are very resistant to change. However, just as the world in which we live in changes every day, the environment that these applications must live in change too. New operating systems, service packs, and even security patches alter this environment. Just imagine allowing strangers to change out pieces of the foundation of your house without consulting you – that is what happens to applications over time.

Microsoft supports the software developers that produce these application by releasing “Software Development Kits”(SDKs) with each version of the operating system, and sometimes with service packs as well. The SDK defines interfaces supported by the operating system and “system dlls”. Independent Software Vendors (ISVs) and in-house developers use these SDKs to create their software. The kits provide the documented library interfaces that the ISV calls to perform many standard functions supported by the OS. These interfaces are to functions that might be as simple as allocating memory, writing a file or to an internet socket, or they can be interfaces to complex operations involving cryptography, ADSI, or SQL.

Generally speaking, Microsoft works hard to ensure that newer SDKs are backwards and forwards compatible. Backwards compatibility, in this case, means that if a vendor builds against the SDK released Server 2003 and Microsoft later comes out with Server 2008, that the software “should” (in most cases) continue to work. Forward compatibility means that the same software would work on an earlier OS (like Windows XP) unless the developer chose to use an interface documented in the 2003 SDK as not available on XP. Forward compatibility issues must, of course, occur or the OS and SDK would never be able to add new features. But backwards are the cause of the painful *application compatibility* issues are the pain we feel in application migration.

You have to admit that on one hand Microsoft has done a fantastic job of supporting backwards compatibility. Unless you run a 64 bit OS, much of the software written for DOS over 25 years ago will still run today. That is absolutely fantastic! On the other hand, Microsoft doesn’t make it very easy for the ISVs to completely change everything that should be changed. Microsoft spends lots of money supporting the development community with the SDKs, Tools, and numerous education efforts aimed at ISV developers. But except for programs like platform ready certifications (e.g.: “Vista Ready”) it is quite hard for an ISV to fully understand what the impact of a new OS or Service Pack on their application code. This leads the ISV to perform the dance called “test and hope”.

And if compatibility issues are not enough, applications also suffer from *application conflict* issues. One application might need one version of a third party component (such as Java) while another application requires a different version of the same component. Try to install both applications on the same OS and at least one of them will break.

The folks at App-DNA have produced a semi-automated tool to help asses and diagnose application compatibility, and application conflict, issues. They have pored over the documentation, blog posts, and knowledge base articles from Microsoft to amass a collection of application compatibility issues. They have built this tool into a platform, called *AppTitude*, that also serves you to manage migration projects of all your applications and understand where both compatibility and conflict issues exist.

Based in the United Kingdom (Longon), App-DNA is a spinoff from Camwood Ltd (http://www.camwood.com/). Camwood is a consultancy with years of experience with application packaging and is probably best known for the AppEditor tool for MSI editing. The AppTitude product began as a side project within Camwood, but the company wisely understood that the product would be unnecessarily hampered if other consultants viewed it as competitive because of Camwood. Spinning off as a separate company frees them to work with all partners on an equal basis and focus exclusively on pushing the envelope on Application Intelligence.

Setup

The product depends upon an environment with some specific tools. You need an Sql Database and IIS server, and typically you will need Virtual Server 2005 SP1. (The company plans to support VMware ESX and Hyper-V in the future). You don't need a lot of horsepower – you will run the host OS and only one VM, so even a laptop with sufficient memory will do.

Any integration with Virtual Server is bound to be very touchy, and this product is no exception. Now I am not a typical customer, as I tend to try things differently than documented just to see if it works. So I had more than my share of issues getting the product installed. For example, you can't install the AppTitude server in a VM. And you can't install it on an x64 OS (due to IIS integration issues). And if you don't have a "C:" drive on the Virtual Server host you will need to manually change a few things to get VM integration working.

Fortunately, the App-DNA support folks were quite knowledgeable on the product and very patient with me. Once we had things set up properly, everything worked smoothly. I would strongly recommend not attempting to use an existing system but to build the software starting with a fresh OS and following their instructions closely.

Overview of Working with Applications

There are three phases to working with an application in AppTitude.

1. Import
2. Analyse
3. Report

Import

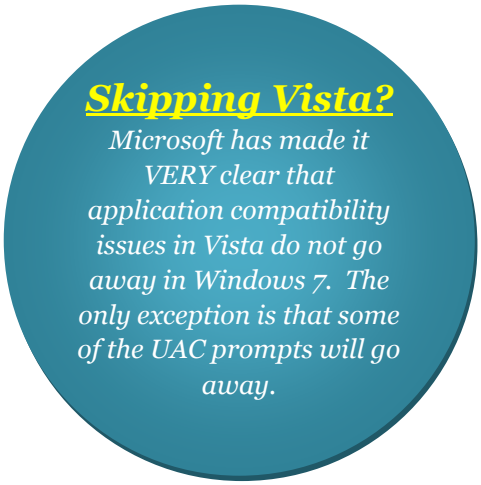
The first phase, “Import” consists of identifying the application to be installed and running the import function. During the import, AppTitude examines the installer and/or installation and collects information into the database. All the important information about files and registry and APIs are recorded here (without analysis).

Remembering that the import is what costs a license, it is good to note that AppTitude has some built in logic that will allow you to import the same application again, replacing the original import. I didn’t test this, but they say that as long as the calculated signature is 90% the same they will match it up.

Analyse

In the second phase, “Analyse”, (the company is British, hence the spelling) the contents of the database for one or more application are analyzed, using a set of customizable rules, to determine suitability for certain kinds of deployments (compatibility) and conflict between applications. These deployment target types are supported through optional (and updatable) modules. The modules I worked with were:

- Interoperability
- Vista Sp1
- XP SP3
- Server 2003 SP2
- Server 2008
- App-V
- (Citrix) Streamed
- (Citrix) Hosted



Each of these modules includes a number of tests (called rules) to be performed against the imported application(s). For example, the Vista SP1 module includes a rule that looks specifically at SP1 related issues, so the module really supports Vista compatibility both with and without SP1.

Another example of a rule is a test looking for calls to what Microsoft calls “depreciated library calls”. You might remember the security push that Microsoft made during Vista. One outcome was that the interface for many memory buffer operations such as to copy a text string changed to help prevent buffer over-runs. The Microsoft SDK continues to support the old interfaces, but considers them “depreciated”. Microsoft has internally removed the old calls from all of their code (or so they say, anyway) and recommends that ISVs do the same. But for backwards compatibility reasons, they did not remove the old interfaces (at least not yet).

AppTitude allows you to customize these rules to an extent. You are not allowed to add new rules, or modify what the rule is testing, but you can exclude rules or modify their impact on the reports. So if you do not care about those depreciated calls (since they don’t break anything today), you can exclude the rule from the analysis. On the other hand, maybe you want this kind of issue flagged in the report as a potential issue, or a blocking issue requiring remediation by the software vendor.

A good example of why you might want to tweak these rules would be in the case of App-V. The module has a rule that looks for use of Microsoft .Net. If I was looking to virtualize the applications using App-V 4.2 (actually called SoftGrid 4.2) I would want such an app flagged as amber, but if I were going to use App-V 4.5 then I wouldn’t want that test to effect how the application is rated.

Report

In the third phase, reports are generated that show the results of the last analysis run. The PDF reports out of AppTitude are absolutely gorgeous!. Print out a sample, hand it to the CIO of your company (or if a consultant your customer) and they will approve purchase of this product (or your service). While the “scorecard” nature of the report looks impressive and will no doubt impress, it is the detail level information that you will need to understand to gain value from using the product.

In the “scorecard” view, each application is rated against a “Red/Amber/Green” scale where Green indicates that the application should transition smoothly to the target environment. Amber usually indicates that there are identified issues that may require a work-around, while Red means that this app is either not suited or will require significant effort. The color chosen for an application in the summary will be the worst case result of any of the rules.

While I can't include the full report to show you just how good it looks, I can pull out a few pieces to give you flavor of the value of the information provided. Below is part of the summary for a sample report taken after a few applications were analyzed. We can see that one app (names omitted to protect the guilty) looks like it is ready to be deployed on Vista right away. A couple violations of "best practice" rules were detected, but those rules were configured to not impact the score as they do not impact functionality or suitability for our purposes. The two Amber applications may need repackaging or a work-around. The two Red applications signify applications that are more risky. In truth, these two applications turn out to be deployable on Vista (and have been deployed by many enterprises), but they include software components that have issues that could be a significant problem. Rather than interpret Red as a show-stopper, interpret it to mean that you probably will need to apply some significant resources to test and possibly modify the application package prior to rollout.

				Vista SP1											
State	Application	Version	Source	Windows Resource Protection	Vista 64-bit	Deprecated Components	Operating System Versioning	Session 0 Isolation	Drivers	User Account Control	Best Practice Violations	Obsolete Components	Environment Settings	Service Pack One	Standard Risk
	[redacted]	[redacted]	msi	0	0	0	0	0	0	0	2	0	0	0	G
	[redacted]	[redacted]	msi	4	0	1	0	0	0	1	3	0	0	0	A
	[redacted]	[redacted]	msi	4	1	9	0	0	0	3	4	0	0	0	A
	[redacted]	[redacted]	msi	152	7	67	28	3	0	24	4	0	0	0	R
	[redacted]	[redacted]	msi	4	7	51	0	0	0	7	6	0	0	0	R

The detail level of these reports are very specific. Every ISV should be running their software through this tool! Here is an example of the detail of one of the tests:

7.3 Deprecated Components

Specific technologies present in previous releases of Windows have been deprecated from Vista. These technologies represent a varied risk of incompatibility.

EXE API calls referencing Critical Section Code

Manifestation: The multi-threaded behaviour of applications making API calls referencing critical section code may be impacted by changes to the behaviour of CSC APIs, however they are unlikely to fail catastrophically.

Remediation: The application should be tested on Windows Vista and may require re-development if the multi-threaded behaviour change impacts the user experience.
More Information: http://msdn2.microsoft.com/en-us/library/ms682530.aspx

So much for the introduction. Let's take a look at these three phases in more detail.

Importing Applications

Once you have the modules you need, AppTitude licensing is by the number of applications you import. You purchase bundles of licenses, and will want to plan out what you import. This import is in relation to the database. During the import, the application installation is captured and parsed. Files are extracted and investigated as needed for later analysis. The results of this are "imported" as information into the database.

The Import phase may be viewed as a fancy application capture. This is accomplished using one of two methods. If the application uses a single MSI installer, it is possible to perform the import by just pointing to the MSI. If the install is not MSI based, or includes multiple MSIs, you will want to use the "Install Capture" method.

Working with MSI-Based Installer Import

When working with an MSI based installer, no virtual machines are used as the application is never actually installed. Instead, AppTitude parses the MSI and content to find what it needs without actually installing it. These are smart people, and they know MSIs – so they can figure out just what would happen without actually installing it.

Working with "Install Capture" Installation Import

In other cases (most, in my opinion), you will set up a job, called a "VM Profile", to be run in the virtual machine. During this job, the installation(s) will actually be run, and at a minimum a difference MSI will be built that captures all of the changes made during the installation. (Note: AppTitude retains the original state of the VM after import processing is complete). This difference MSI is then used for the import. This method supports not only non-msi installs, but multiple installs as a single package. It supports both automated install scripts and a manual mode where you install things by hand in the VM. In addition, AppTitude provides some additional add-ins you can insert into the job (VM Profile).

- **OsSnapshot**. This is a base piece that you need. It operates similar to the old Windows 2000 "windiff", taking a snapshot of the OS (inside the running OS of the VM) before and again after and generating a difference file. App-DNA actually outputs the difference in the form of an MSI. While in theory you could use this MSI to install on a blank machine, it really isn't adviable to use this as an application re-packager as the differences are not scrubbed to remove unwanted capture. All of the AppTitude analysis will be done using this MSI.

- **Security_diff**: This is an optional extension to the OsSnapshot tool that extracts security related differences, DACLS and the like, on files and registry. Although the tool captures the information no processing is done by the tool and you have to read the output file. If you are concerned and want to know exactly what security related changes were made during the installation and/or runtime of the application this tool will tell you.
- **Scrncap**: A screen capture facility. This runs in the VM and helps automate the creation of documentation on how an application is installed. At any point of the install, just click a button and give the picture a name. Although the capture occurs on the VM display, the file is automatically saved to a folder on the Virtual Server host. It also generates an html file that has each of the snapshots with a timestamp of when the snapshot was taken.
- **Simplefilemonitor**: The simplefilemonitor captures the actual files changed and copies them into a folder structure on the Virtual Server Host. Like Security_diff, this tool only performs a segregated capture for your review. It does not seem to be anything that isn't in the MSI, but is a more convenient format for reviewing.

In working with a number of applications, I discovered that while my instinct is to fully customize the application during install so that what is captured and analyzed is exactly what will be installed on the target, in practice this seems to be largely unnecessary if you are just using the OsSnapshot tool.

Analyzing Applications

After importing, you can run the analyzer so that you can later produce a set of reports. As you import additional applications, you can run an individual analysis and report on that application only for compatibility, but you can also run multiple application analysis and reporting to look at a potential deployment of a group of apps for both compatibility and conflict.

Analyse will run a secondary analysis on one or more imported apps in the database, based upon purchased and configured "modules" and produce a report. In talking with Paul Schnell, CEO of App-DNA, he points out that one advantage that App-DNA derives from this two stage analysis is that the customer can normally upgrade to new or improved modules and immediately gain the benefits without re-importing the applications because they record all the potentially interesting information into the database during import. So in theory, you only import an application once. Even if the AppTitude releases an update to the module rules, or creates whole new modules (do you think you'd want a Windows 7 module sometime down the road?) you *should* not need to import the application again unless the application itself is updated.

These modules are in essence potential “targets” that you might be considering deploying on or with. The modules I worked with were:

- Interoperability
- Windows XP SP3
- Windows Vista SP1
- Windows Server 2003 SP2
- Windows Server 2008
- App-V
- Citrix XenApp Hosted
- Citrix XenApp Streamed

Although you can set-up your test VM to use the appropriate deployment operating system, this is not necessary. Similarly, you can include App-V Sequencer or the Citrix XenApp Profiler and run the installer through it for the import, but doing so will not improve your import or analysis (and may have the opposite effect).

Bottom Line

So how good were the results? Not perfect, and perhaps a bit over-dramatic, but even I learned a few things about the applications I ran.

For example, I found an app that I thought was perfectly safe, only to discover that it would have a problem on a 64-bit Terminal Server because it has a 16-bit dll I had never noticed.

But I also had a home grown application that I know has a problem under Vista, but it passed the analysis with flying colors. Why? Well the app was originally written for UNIX, ported to Windows in the Windows 95 days, and last rebuilt as a multi-threaded GUI app for Windows 2000. I haven’t touched the source code since 2001. It happens to use an API that was valid on 2000 but even then was “depreciated”, and it seems to have been broken under Vista. I don’t know if this is a case of Microsoft failing to document, or if because the depreciation occurred before Windows XP and thus App-DNA missed it, but the bottom line is the app doesn’t work and I would not know until I actually test the app.

The results can appear to complain about things that don't matter a bit. But that is the nature of what we are asking for. It helps to have a bit of experience to be able to correctly interpret the detail level of the report. And of course you can then adjust the module configurations and re-run the analysis and report to obtain a more suitable looking result.

Ultimately, the bottom line is that if I had to migrate a list of applications, I could save a significant amount of time and energy by having this information first. AppTitude is a great example of applying automation to repetitive tasks requiring excruciating attention to detail to make life a whole lot easier.

Post Notes / Summary

It would be really great if App-DNA was an industry funded non-profit company. Make one central database and let people download the tool and upload results. Then we could all just run a web report and get the info we want. Maybe Steve Balmer could donate some of the money he didn't spend on Yahoo? I guess that probably is not going to happen.

It would also be really, really, great is every ISV used this tool as part of their development and test process. App-DNA doesn't really have a pricing model that works for them today, but I'm sure that if someone was interested they could work something out that was fair.

AppTitude is pushing the envelope on compatibility and conflict analysis. It does not really compete with, nor replace, the "Application Compatibility Toolkit" that Microsoft has. There is a need for both. I hope App-DNA continues to push the envelope on Application Intelligence for years to come.

Although an evaluation product and technical assistance was provided by App-DNA, this paper was not paid for by App-DNA and represents a fair and independent technical evaluation.

Tim Mangan an independent consultant with the title of *Kahuna* at TMurgent Technologies LLP (www.tmurgent.com) based in Canton, MA USA. Tim is *a Microsoft MVP* for Application Virtualization and his company provides training and consulting in App-V as well as other virtualization areas. Tim may be reached at tmangan@tmurgent.com .