PhUSE 2012

Paper RG01

New Approaches to Validation for SaaS-based Clinical Computing Solutions in the Cloud

Tony Hewer, Medidata Solutions Worldwide, Uxbridge, UK

ABSTRACT

Software as a Service (SaaS) is evolving as the preferred model for many life sciences companies. Moreover, operating such solutions in a Cloud environment is also emerging as a viable – and, in many cases, a preferred - option. These new paradigms are challenging the traditional approaches that have been taken to the overall validation of such computerized system solutions.

This paper provides a high-level introduction to SaaS and Cloud Computing and outlines a revised approach to validation that enhances development and operational efficiencies whilst still maintaining the key requirements and expectations of sponsors, regulators and other stakeholders in the highly regulated environment in which we operate.

INTRODUCTION

To quote: Medidata Solutions is a leading global provider of SaaS clinical development solutions that enhance the efficiency of customers' clinical trials. Medidata's advanced solutions lower the total cost of clinical development by optimizing clinical trials from concept to conclusion.

In this paper I am going to cover a number of inter-related topics – all related to this paper's title. Whilst the topics might appear disparate, I trust that they will bind together in the conclusion. Specifically, I shall cover:

- Key concepts
- Working in the Cloud
- MIST and SIMT
- Agile software development
- Computer system validation of such software
- Regulatory compliance of such software
- Customer UAT
- Audits of providers.

KEY CONCEPTS

The Cloud! For the sake of clarity – and simplicity – let's split the cloud into three layers.

Starting at the "bottom" with IaaS – Infrastructure as a Service. This is the hardware in a data center – the servers, the data storage devices, the local-area and wide-area networking devices, the load-balancers and so forth along with firmware and so forth.

"Above" that comes the PaaS – Platform as a Service. This comprises operating systems, database systems and other "middleware" software that can, to a very large degree, be regarded as ubiquitous and highly standardized.

Next up – and on the "top" - comes the real business value-facilitating components, the SaaS – Software as a Service. This is, typically, application software that takes care of (i.e., processing, manipulating etc) data!

One key aspect of a Cloud-based solution is its elasticity; that is, it's ability to readily scale-up (and down) in proactive response to demand - largely driven by key parameters like the number of end-users or the numbers of datapoints having to be handled or the number of transactions being processed. This scaling must be catered for at all three of the SaaS, PaaS and IaaS levels.

Agile! You might wonder why this is included here; surely this is a topic in its own right and has nothing, per se, to do with Cloud Computing or SaaS. Empirically, this is, of course, correct, but I'm am including it simply because the very nature of Agile generates virtually continuous change to the SaaS level; certainly to a much greater, dynamic extent than happened when new releases of application software happened very infrequently.

This dynamism does, in itself, stress the SaaS layer – and, hence, potentially the PaaS and IaaS too. So, in other words, Agile is good for SaaS! More on this below too!

MIST AND SIMT! These acronyms stand for Multi-Instance/Single-Tenant and Single-Instance/Multi-Tenant respectively and, essentially, pertain to the architectural nature of a SaaS component. The SaaS layer can therefore comprise of multiple instance of MIST components and single instances of SIMT components. A simple example from Medidata is shown in Figure 1 and, importantly, illustrates that MIST and SIMT can co-reside in an overall SaaS-based solution.

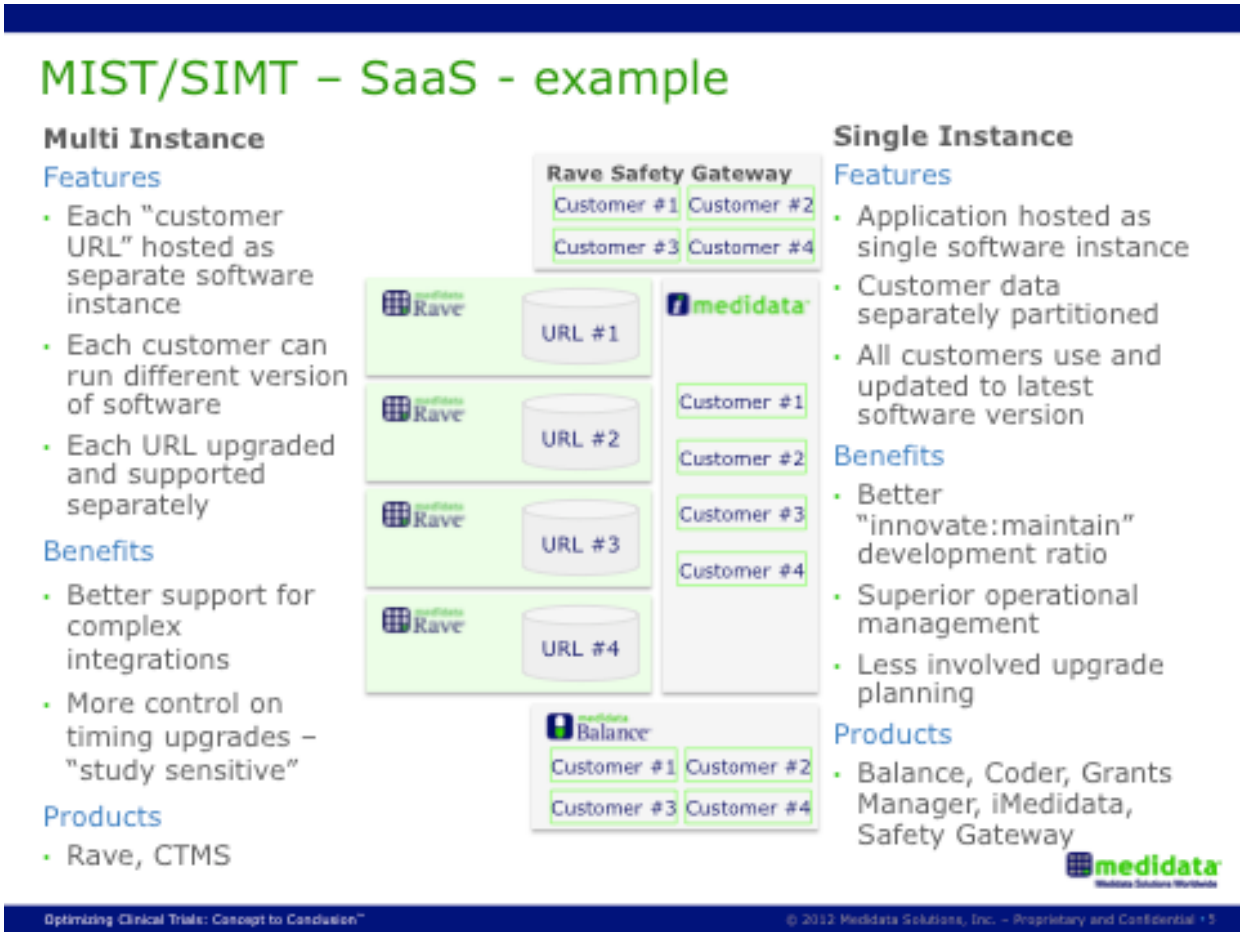

Figure 1: MIST and SIMT example

Computer Systems Validation! With this IaaS, PaaS and SaaS model, what does this mean in terms of validation of "the computer system". Indeed, what *is* the computer system? All in all, some interesting paradigm shifts are necessary which I'll deal with later.

The regulatory compliance angle of software in this SaaS model is, actually, straightforward and conventional. Put simply, the application software must perform its duties so that it enables its users to be compliant. The approaches taken for designing, building and testing that software therefore have these regulatory requirements as key requirements. No change really from the "traditional approach"! That said, the Agile approaches do lend themselves to – and, indeed, drive the need for - much greater use of, for example, automated testing.

Customer UAT is an old concept and is generally regarded as a pre-requisite to using software in a "production manner". The SaaS model challenges the extent and nature of the UAT. Again, I'll talk more on this below.

And finally, there is the matter of audit-like scrutiny of providers of SaaS and, indeed, PaaS and IaaS. There have been long-established regimes for software providers in our industry getting scrutinized. But, again, a new approach is needed; again, I'll deal with this later.

WORKING WITH THE CLOUD

Let's look at this both from an ongoing, production aspect and, separately, what the development and maintenance looks like.

I've already touched upon some core attributes of working SaaS solutions in the Cloud above but let's examine the following list:

- Scalability: The elasticity afforded by the IaaS and PaaS "layers" can be leveraged by the SaaS layer. The application software does, of course, have to be cleverly and elegantly designed to facilitate this.
- Availability: IaaS and PaaS providers distinguish themselves by providing continuous service with almost zero service interruption. This is facilitated by super resilience of their components across, typically, multiple centers of operation with such resilience also in the complex networking that's inevitably needed in-center and inter-center.
- Reliability: Same paradigm!
- Security: Again, the providers must distinguish themselves in this space both at the physical and logical levels.
- Self Service: One key aspect of SaaS is that the user is more and more empowered to configure and control their use of the service. For a SaaS provider like Medidata such self-service is, for example, provided at the end-user profile setup – or the clinical study setup.
- Configured: The software in the SaaS layer is the same. Client-specific aspects are handled thru *configuration* – not *customization*.
- Up-Versioning: The SaaS provider can update components very frequently – fixing defects and introducing new features. For SIMT applications, this means that all clients automatically get that new version all at the same time. And, for MIST architectures, it would be desirable if this adoption model were like that too so that the SaaS provider can be focused on support for as few versions as possible.
- Standards: The adoption, as much as possible, of standards within the software of the SaaS-based solution is to be encouraged as much as possible. By "standards" here, I'm referring to both internal as well as external. With Medidata, as you might expect, we have "standards" within our software base as to how data gets stored, exchanged etc. So far as external standards are concerned, we also can talk ODM and other languages.
- Inter-operability: A downstream of standards adoption facilitated by publication of our standards.
- Development efficiency: See more below about this.
- User Experience: The big benefits here are around consistency and a fantastic user interface making the software a delight to use. This has, of course, always been a challenge for software companies but today the nature of software development revolves around that user experience. Putting that user experience into a SaaS solution in the Cloud makes it ever more important and a huge differentiator.
- Lower Risk: The SaaS model promotes the ethic that that the application is ubiquitously reliable, available and meets its requirements. The risk focus is therefore shifted toward the areas of the client's use of the application in the context of his business model. And it is therefore this risk that be the focus of his testing. More on this later.

For the maintainers and developers of the SaaS software, new approaches have been developed around:

- Software Development: This is highly geared toward rapid, incremental development of software that's to be deployed (as a SIMT or MIST application) frequently.
- Software testing: This has become highly automated. In fact, it has also become continuous. The whole code-base can, in effect, be fully regression-tested after every change. The art – and science – of doing this has become extremely streamlined. But what it does mean (very vividly) is that the old-fashioned approach of "hand-written" test scripts and user-driven test script execution (often using pen and paper) has [almost] become a thing of the past.
- CSV: The fundamental principle of CSV – that the system should do what it says it will do – still stands. The paradigm shift is how this gets achieved. Consider the notion that the SaaS provider provides a solution that is declared as *validated* or, perhaps, more accurately, is in a *validated state*. The SaaS provider is accountable for sustaining this *validated state* at all times – especially when it fixes defects or updates versions. The client must have confidence in the provider that the procedures and evidence associated with such ongoing activities do, indeed, sustain the validated state. What the client does after each defect fix or upgrade is up to him in the context of *his* risk. More on this below.
- Security and Privacy: Whilst there have been number of high-profile and lurid instances of security breaches in recent times, I would assert that client information (of whatever kind) is probably more secure than its ever been by virtue of IaaS/PaaS/SaaS adoption in the Cloud. The providers have it as a key goal and if they cannot live up to it, they will not survive for long. Before, security was only as good as you (the clients) could make it – or could afford to make it! This is, of course, a rapidly moving target but the providers are far better positioned to stay ahead of this curve than users or clients could hope to be.
- Deployment: This is fast and furious. And, moreover, it's easily done by the SaaS provider who treats the IaaS and PaaS as services, not having to care (too much) about the complexities of server configuration, networks and so forth.

AGILE

I'm only going to deal with this topic superficially in this paper. Please refer to the 2011 PhUSE paper that I co-wrote and co-presented with my colleague Andrew Smith for more information about how Medidata has adopted Agile as a pivotal technique for bringing our SaaS-based offerings to market.

The main reason for including it in this paper is to highlight the point that we could not regard ourselves as a true SaaS provider if we were not using Agile. Or, thinking about it a different way, by using Agile, we can think of no other model to adopt other than be a SaaS provider – certainly for the kinds of products and services that Medidata provides.

Let me briefly draw out a number of key aspects of our Agile approaches that are pertinent to this paper:

- Each software release has a focused team representing all disciplines of the company from development through to hosting – all working together to bring about a defined result
- Iterative Development – Development is broken down into chunks that can be focused on in smaller timeframes
- There are checkpoints between these chunks to better access what is being delivered and insure that what is being asked for is actually what is really required.
- Agile leverages test driven development whereby tests are introduced as code is written, thus ensuring that whatever requirements are introduced, testing is guaranteed.
- Agile also takes advantage of automated testing; automated testing, as mentioned previously, has the advantage of being able to run tests on an ongoing basis.
- Regulatory continues to have a role in ensuring that the resulting release meets the ever-changing regulatory requirements and that the development and validation process still produces deliverables that are expected by the regulators.
- Customer Focus – The Product Manager owns the business requirements and WILL be in touch with customers along the way to validate assumptions and or help with testing. The Product Manager is accountable for the product release's ongoing success.

CSV – COMPLIANCE VS QUALITY

The "traditional" approach to CSV has tended to center around the production of a documented validation package. That package – again, traditionally – has consisted of a small mountain of documentation associated with testing scripts, specification documentation and so forth. And considerable energy was expended on ensuring that the package was complete, consistent and "pretty".

But, importantly, did this approach give us software that was low on defects? And, even more importantly, did it give us software that was "required"? The answer is anecdotally, a big No. The focus has been on Compliance [with procedure] and not upon Quality.

In Medidata we have adopted paradigms to focus on Quality [of the SaaS deliverable] – not at the expense of Compliance but as the primary facet of the validation package.

SaaS Provider CSV and Customer UAT

It's not just about producing a pretty validation package….

medidata
Medidata Solutions Worldwide

The key to achieving this paradigm set is mutlti-faceted:

- Set the requirements. This is NOT a one-off exercise but, instead, is part of the incremental approach baked into the Agile techniques.
- The Software QA function is focused upon developing and executing tests around these requirements. In fact, the software development itself is driven by the testing regime. The SQA experts strongly influence the nature of the software – especially around matters like edge-cases and regressive functionality.
- Compliance – in the sense of having documented evidence that procedures have been followed comes for nothing! The evidence is, in the main, all held in systems (rather than on paper). Examples are:
    - Requirements Specifications take the form of User Stories in the product backlog or, in the case, of ongoing work, in the development sprints
    - Test Scripts take the form of user scenarios within Feature Files
    - Test evidence takes the form of logs of automated test executions; these can, in many cases, contain embedded screen shots.
    - A traditional Validation Plan is still used as a highly valuable document but is one the few "traditional" documents produced.
    - Similarly, the Validation Summary and Validation Certificate are also produced as traditional documents and we often make these available to clients as attestations of our validation efforts and compliance with regulatory requirements.

CUSTOMER UAT OF SAAS-BASED SOLUTIONS

So, if we're doing all this great stuff around validation of our SaaS in the Cloud products, what's this mean for the clients?

A principal focus I would assert is that the client has to be sure that the software supports his requirements to be compliant with applicable regulations. Key here is regulations in the areas of GCP, ERES and data protection/privacy. So how does the client check this? He checks his provider; see next section.

In addition, does the client really need to fully validate the SaaS solution? Answer: No! Instead, the client should adopt the approach of developing and executing a package of UAT focused upon their intended use of the

solution. That could be a broad set of tests focused upon their assessments of risk in the context of their business operation.

We can all agree that the manufacturing of drugs, biologics, and medical devices is very critical – and that the systems used in manufacturing operations need to be extensively tested. The risk is high because a bad product could result in a very negative impact on the patient.

In the clinical arena, there is a range of operations. No one is saying the monitoring trip reports aren't important, but the risk to patients if there is a minor malfunction in preparing a trip report is pretty low. So, for instance, a company might take 3-6 months to test a particular Manufacturing module and yet only take 1 day to test a monitoring report. This is reasonable.

You should also consider how widespread the use of the system is within the industry. And is one of the advantages that of using a commercially available SaaS solution? If there are many customers using a SaaS/Cloud based solution, there is a higher likelihood that if there are problems, that these problems will be detected – and fixed. And they will be detected more quickly than if a client is using an inhouse developed system with a much smaller set of users.

In addition, we should remember that regulations and guidance (for instance, what we see in GAMP, ICH, US FDA) –not only accept a risk-based approach but, in fact, encourage it. This applies not only to validation, but also things like risk-based monitoring and targeted source document verification.

So, in this SaaS/Cloud environment, we encourage clients to take a much more flexible and innovative approach to UAT of the systems. Clients should focus on how they have *configured* the system for *their* particular use. Focus on integrations with other inhouse developed systems or applications from other providers. It is these configurations and integrations that will most benefit from a focused UAT. Take advantage and consider the large amount of testing that the provider performs. Clients should actively participate in any pre-release testing of new versions of solutions – especially pre-releases of SIMT applications.

It is Medidata's opinion that clients shouldn't validate or retest the entire functional set and code base of the system. Ultimately, this is your decision to make; we have some very conservative clients who want to redo all the testing that we have done. But it seems that client efforts could be better spent elsewhere, and clients would be able to take advantage of new enhancements in a much quicker manner.

SCRUTINIZING THE SAAS PROVIDER

There is, of course, a regulatory expectation that clients scrutinize the suppliers they engage – in the context of this paper, this means the SaaS provider. And, indeed, the IaaS and PaaS services utilized by that SaaS provider because that touches upon the important areas like availability, reliability, performance and so forth.

Clients should ask their SaaS provider how they are conducting their own assessments of their IaaS and PaaS providers on whom they depend. Are there 3rd party certifications such as SAS 70 (now SOC 2) or ISO or CMMI? Medidata has a formal process for conducting such assessments of such suppliers. And, in addition, we have achieved our own SOC 2 certification in our own data center hosting operation. In many cases, the resulting certification reports are available for customer review.

Clients should follow-up to see how Disaster Recovery and Business Continuity are defined and tested. It is also of critical importance that the provider is able to ensure that client data is secure and kept private.

Examine the development methodology used by the provider. As we have discussed, Medidata has moved to a development approach that is truly designed to improve the quality of the product, and not simply result in a "quality" validation package. Clients should look at the breadth and depth of automated testing. I have explained how automated testing is of real benefit, because tests can be run continually, thus helping to ensure that new problems are not introduced.

Traditional QA audit focus relies heavily on looking at the provider's Quality System, compliance with procedures, training records, etc. I am not saying that this traditional focus isn't important. But Medidata's experience – and we host perhaps 70-100 onsite audits every year – is that the truly valuable client audits are those that focus more on some of the other topics we have discussed. And not simply on those QA type activities that QA has been focusing on for as long as audits have been occurring. So, should the focus of clients during provider audits come from business representatives [rather than – or in addition to – the client's QA function]? Can consideration be more strongly given to audits of providers by a client consortium? Can some form of independent, 3rd-party certification of the provider be of value?

And finally, be careful of the square-peg in a round-hole. Clients' procedures for assessing SaaS/Cloud providers may need adjustment; many client procedures that Medidata has encountered are geared toward inhouse applications or suppliers of licensed software and this model is not appropriate.

CONCLUSIONS

In conclusion, let me summarize (in slide form), the key messages from this paper:

- **Cloud SaaS solutions** – client can focus on their core business (drug, biologic, device development; contract research services)
- **Cloud PaaS and IaaS solutions** – SaaS provider can focus on their core business (innovative software development)
- **Single Instance Multi Tenant (SIMT)** – more timely adoption/implementation of software updates
- **Agile software development** – improved product quality, introduction of enhanced features in a more timely manner
- **Customer UAT** – focused on unique aspects of customer's implementation (configurations, integrations)
- **Audit models** – customer can take advantage of alternative audit options

RECOMMENDED READING

Computer System Validation – Is it Fit for Purpose; Ron Fitzmartin and John Wise; Touch Briefings, 2011

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at thewer@mdsol.com.