

iTKO Overview

Brian Davis
iTKO
Jan. 13, 2009

iTKO  *LISA*™

1

© 2008, Interactive TKO, Inc. PROPRIETARY / CONFIDENTIAL

WWW.ITKO.COM

I'm going to present a brief overview of iTKO and our LISA solutions.

iTKO LISA™

iTKO: Our Customer Mission

Enabling SOA Agility & Quality

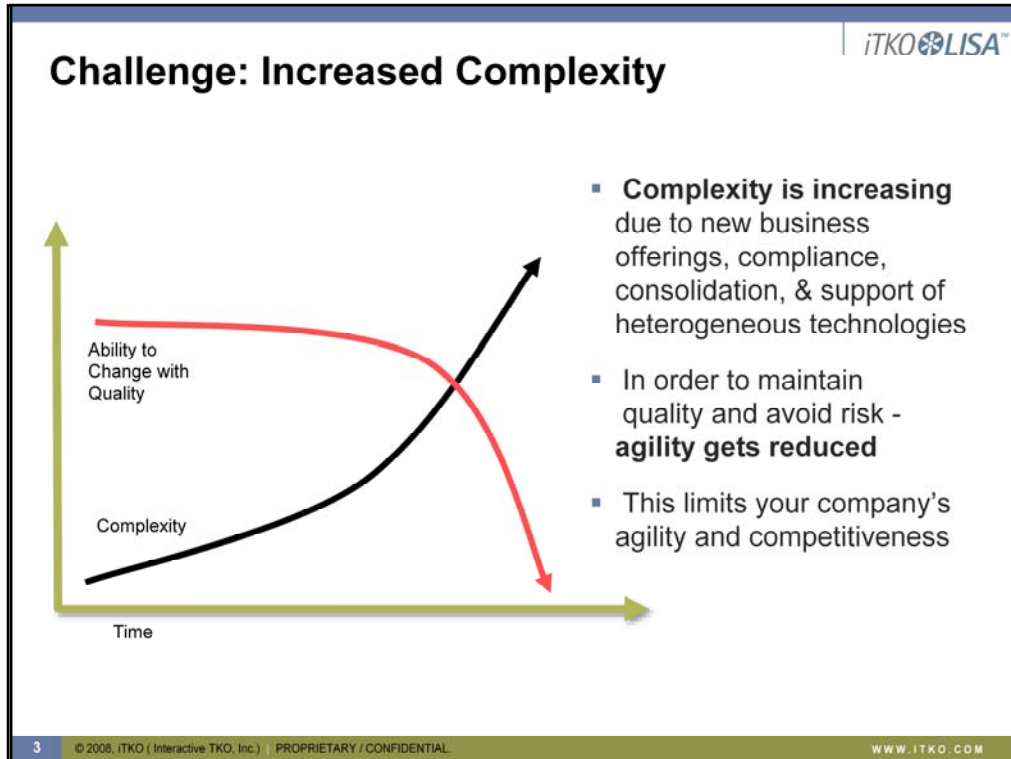
in today's complex technology environments

2 | © 2008, iTKO (Interactive TKO, Inc.) | PROPRIETARY / CONFIDENTIAL | WWW.ITKO.COM

iTKO was founded in 1999 to Enable SOA agility and quality. Yes service oriented concepts have been around since even prior to that time and we have been in the business of helping companies like these reap the rewards of service-based architectures since then. Our product was launched commercially in early 2003.

Customers come to us to help succeed specifically in SOA, but others come to provide virtualization of legacy systems, or provide quality and continuous validation for their ESB initiative, and other reasons.

From here, just identify with the customer's similarity to another customer...

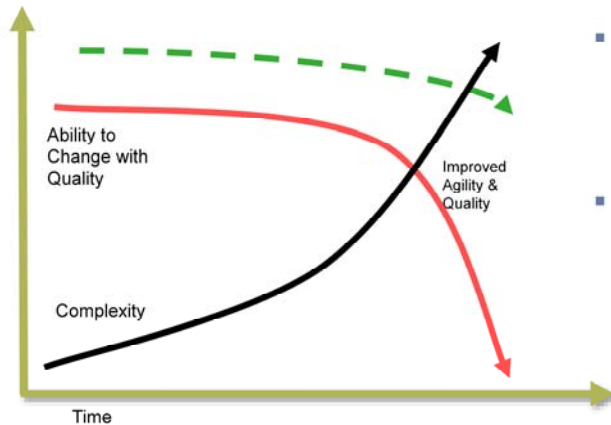


Today's applications are getting more and more complex. SOA, ESB, Composite applications, and N-tiered architectures all contribute to this technically. The business drivers of M&A, automating business partner transactions, BPM, and outsourcing cause the business to need more interconnectedness & speed of delivery.

It follows that as you increase the complexity of your environment and hold all other things constant your quality will suffer.

To avoid this, most organizations have to slow their pace of change, threatening their agility goals for business agility

The Goal: Enable Agility & Quality



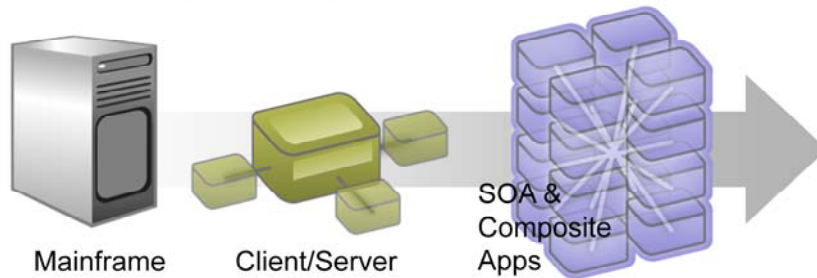
- Increased Revenue due to faster time-to-market and reduced customer failures
- Reduced costs and increased efficiency of development and testing

By adopting best practices and technology support that is enlightened to the challenges of agility and quality on complex systems, we can improve these things and bring down costs. We can also increase our development throughput and therefore revenue to the business.

What I'd like to do is introduce you to some of the issues standing in the way of Agility and Quality, then show you some best practices and tooling to resolve them.

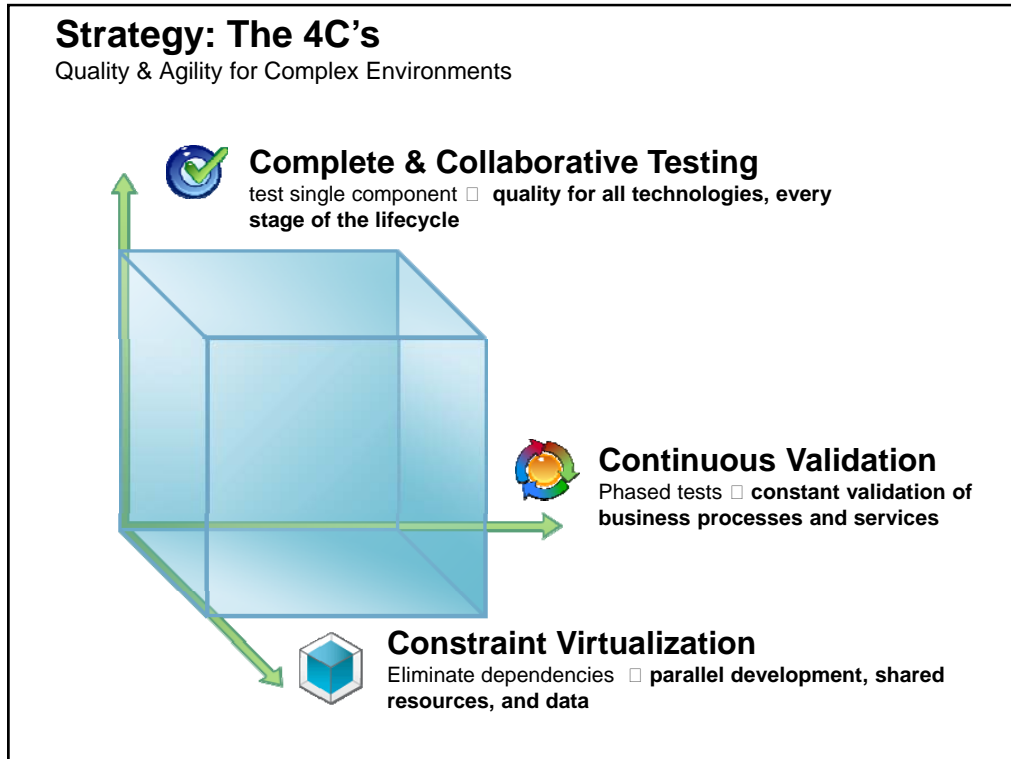
Specific Issues to Resolve

- **Inadequate Testing:** Traditional testing techniques cannot automate testing against component and middleware-based workflows
- **Unintended Consequences:** Individual components and technologies are changing at their own pace and can create severe issues within business processes
- **Dependency on Constrained Systems:** Access and capacity constraints on key systems greatly reduces agility



Here are some of the common pitfalls that need to be addressed so that you can enable agility & quality and decrease the cost.

(now just name them)



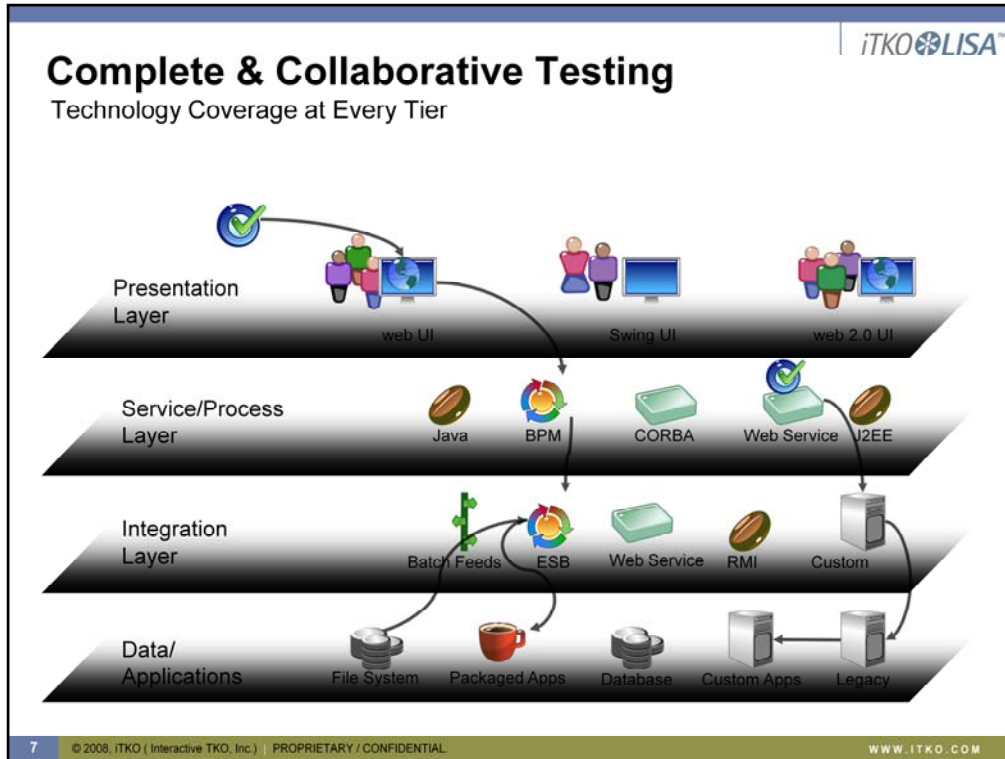
We have been doing this a long time, and we have been working with customers for years on how to get these service-based architectures to deliver on their intended value. We have distilled that domain expertise into a best practices framework we call the 4-C's for Quality and Agility.

My assertion to you is that if you get these concepts into your strategy you will dramatically increase your chances of success.

This is true regardless of how, but of course I'd like to 'cut to commercial' as I discuss each of these areas to address how our product LISA can help you accomplish these.

At a high level, you can see there are 3 areas of concern, Testing, Validation, and Virtualization. These are the enablers for Quality and Agility.

Name the C's if you want, but DO NOT describe them here!!!



Complete testing simply acknowledges the fact that we are in a heterogeneous environment and that this causes our test strategy to become equally heterogeneous. It is no longer safe to simply assume that we can test at the UI of an application and assume all is well. We may not even have a human interface at all – it might be a machine interface like a web service we need to test.

Yet we can't simplify the problem to any one layer like the web services layer either. We have to understand that our not complete unless they can invoke logic at one technology layer then verify at a deeper level. We call this "invoke and verify". Without this we are implicitly trusting the very system we are testing.

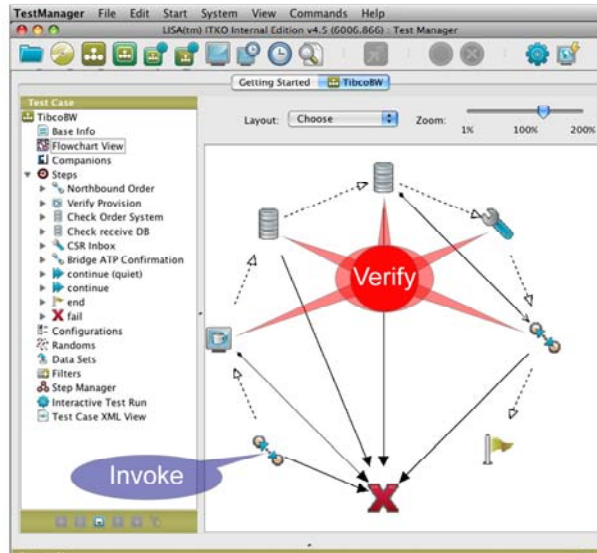
For example at a Bank ATM, you might 1) make a deposit, 2) look at receipt, 3) but the receipt is not sufficient to prove outcome, 4) must verify "around" the ATM at the mainframe or call center app or something to ensure that the balance was actually credited to your account.

You won't be surprised to hear that LISA is really good at this. Our technology coverage is unmatched. We cover all the commonly used standards in service-based architectures and have native integration with most J2EE servers and ESBs. Our extensibility is strong and helps us cover even your proprietary technologies so that testability issues can be resolved.

Complete & Collaborative Testing

Invoke BP Model, Verify Outcomes

iTKO LISA™



8

© 2008, iTKO (Interactive TKO, Inc.) | PROPRIETARY / CONFIDENTIAL

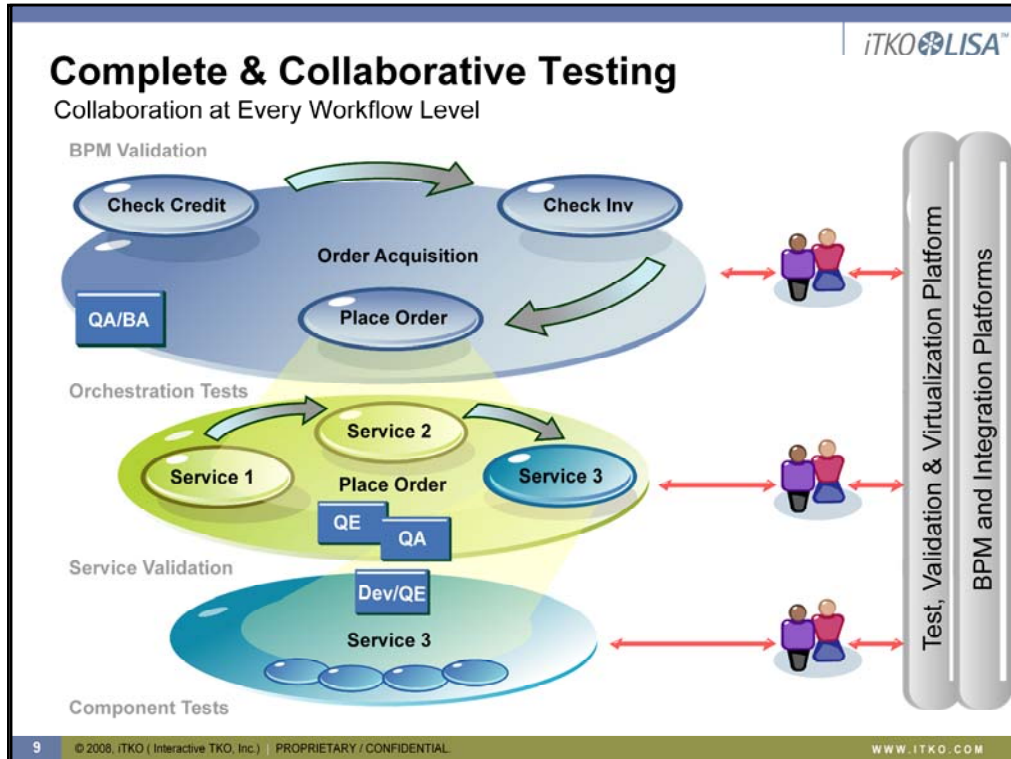
WWW.ITKO.COM

Here is an example of testing a business process model. To do this LISA is used to invoke the business process, then verify the expected outcomes. Invoking might be using a UI, making a web service call, putting a document on an ESB, or even dropping a file into a drop-box folder.

Verification is equally heterogeneous... Did the ERP system get the order message with the right customer data?

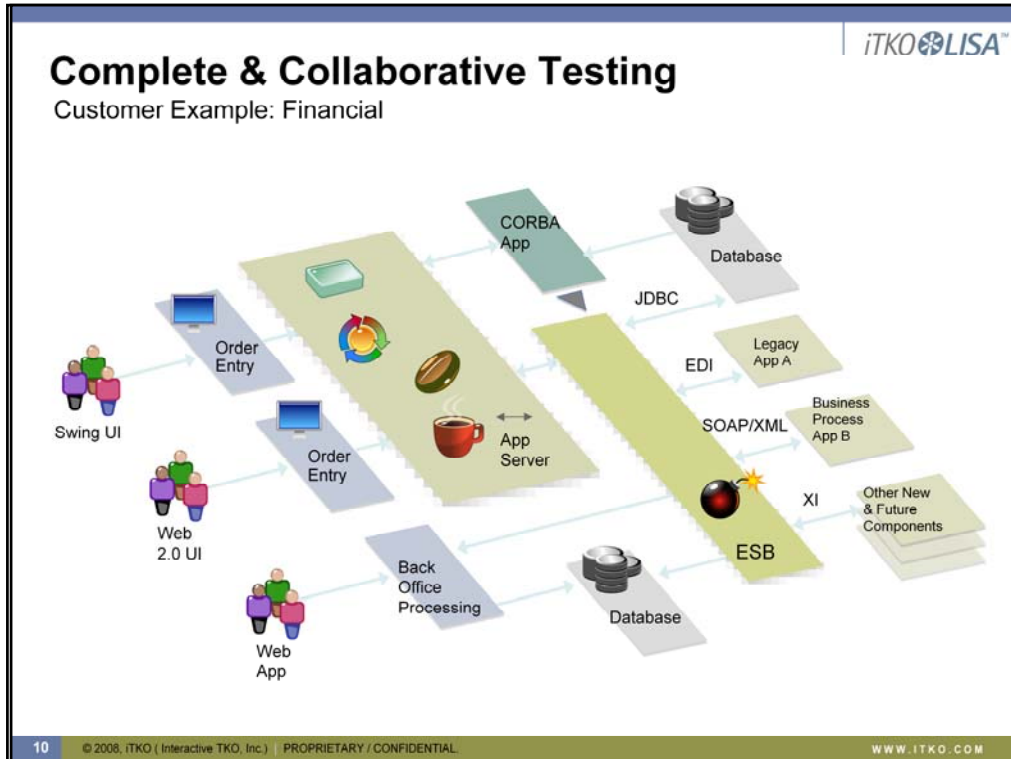
Did the customer billing system see the update?

Did the shipping queue get its message?



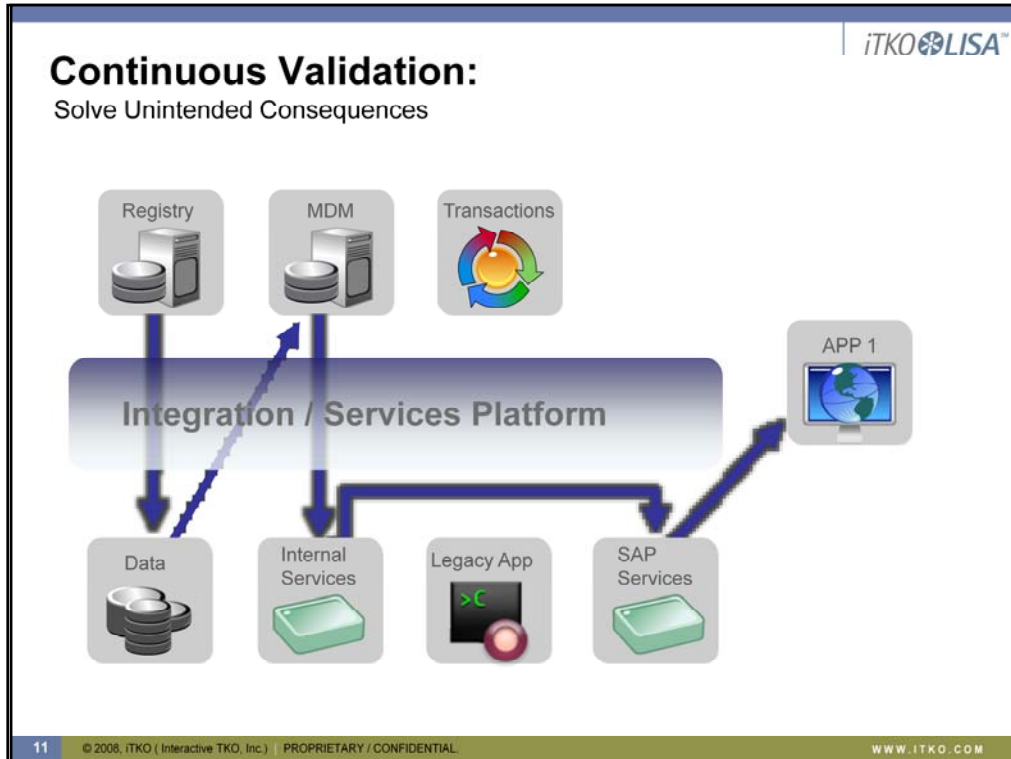
Collaborative is our next C. It acknowledges the fact that enabling agility requires more parallelism and shared effort and data in our delivery teams. To do this we need tests that start from development and can be shared through the lifecycle. That means unit or service type tests can be consumed by QA, QA can iterate on those tests and bounce back failing scenarios. Appropriate tests can be used as performance tests. And even into integration and production.

The other area in which we must collaborate better is to provide visibility. Just like we model business processes with sub-processes to encapsulate, we need the same thing here. This way QA and business analysts are not expected to understand every technical nuance of these complex systems and yet they wield a lot of testing power on them.



Here's an example of a customer that has the classic "Complete & Collaborative" testing problem. Robotting at the UI was not helping them uncover the issues in their system that were occurring within the Message bus and underlying systems over a period of months. All of the contributing teams of these heterogeneous technologies need to be involved.

(listen to the story)

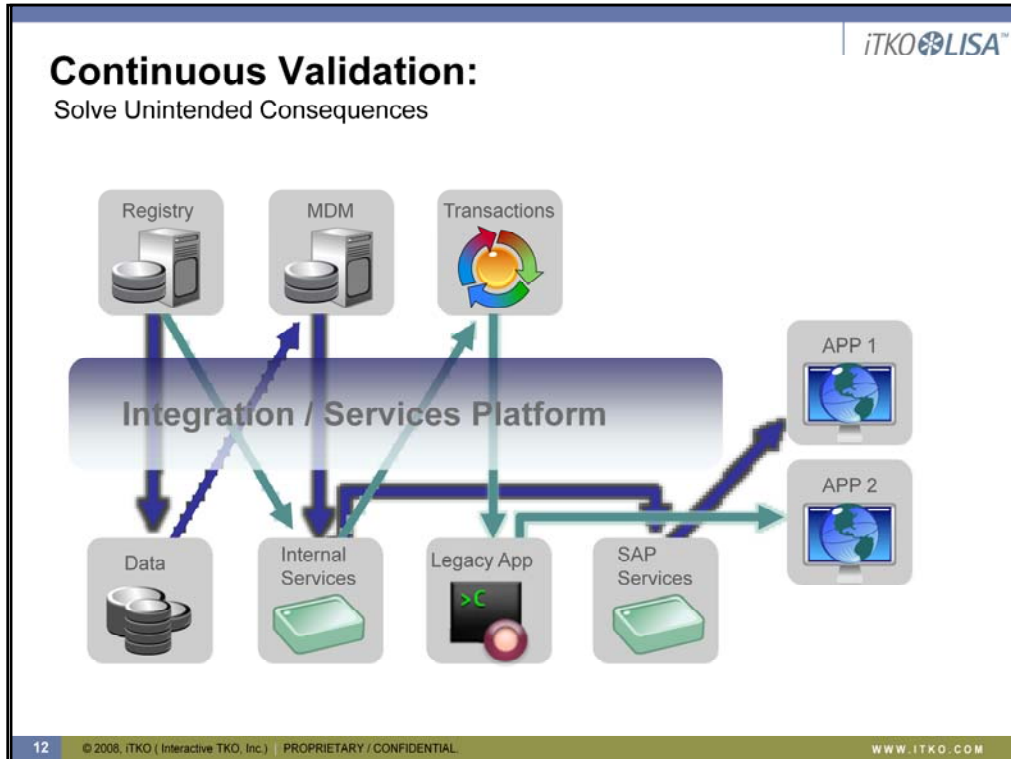


The 3rd of our C strategy is Continuous Validation. We must acknowledge the fact that systems can no longer be ignored from a quality point of view once deployed. If we do, we threaten quality and therefore agility as unintended consequences will catch us off guard.

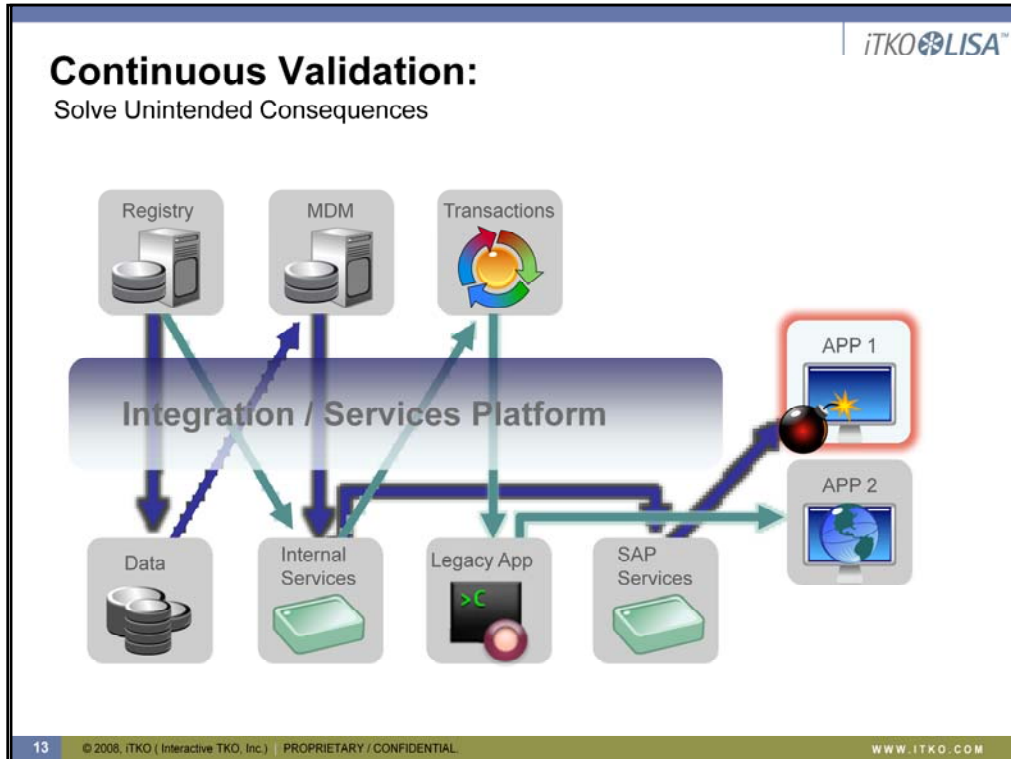
If you have time, give the plumbing analogy

Here's scenario:

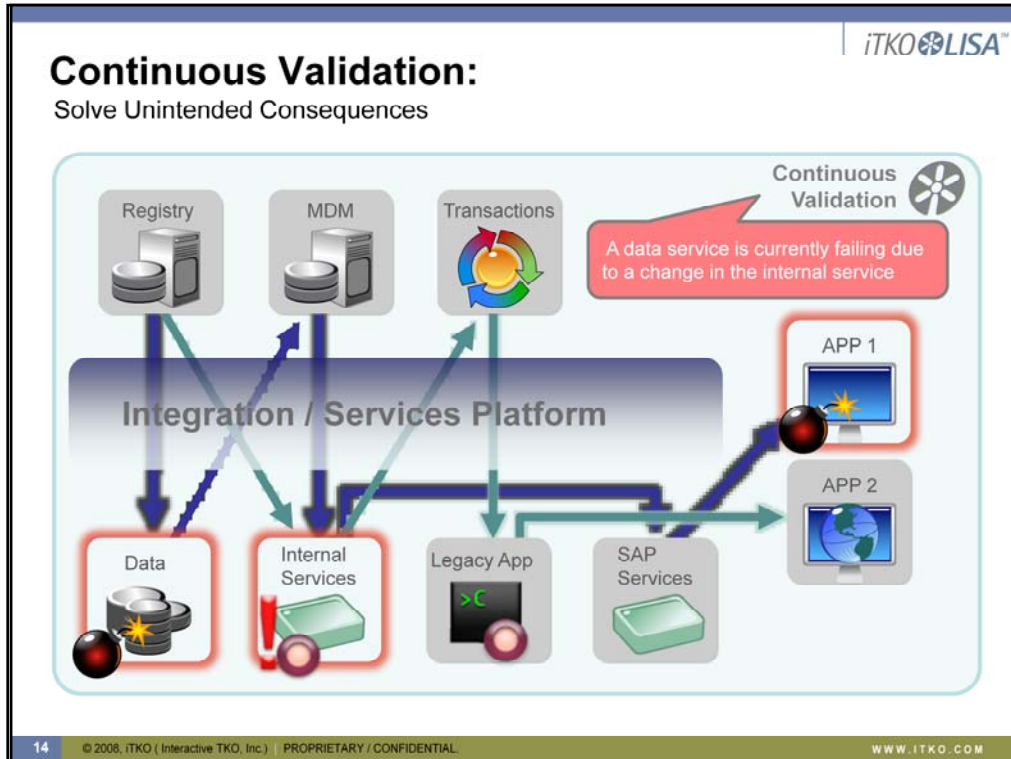
The service in the picture below were developed in support of App 1...



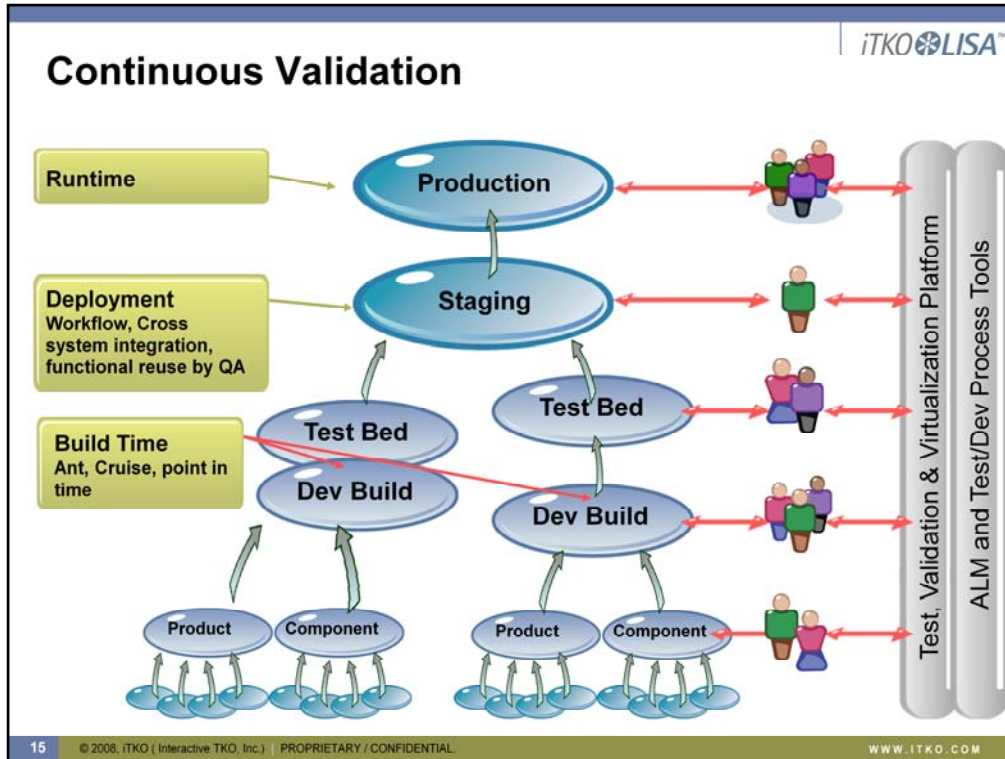
Then a new application 2 arrives and starts consuming some of those same services. In support of this, existing services are reused and some are enhanced. Sounds like we are winning on this whole SOA thing!



But OUCH, once App 2 hits production App1 is now failing. After taking way too long to realize, the App 1 team realizes that a data service is no longer working as expected. So guess who's upset? But wait, the data service wasn't even changed! So it turns out that an internal service used by the data service is now subtly different. That difference is causing an upstream failure.



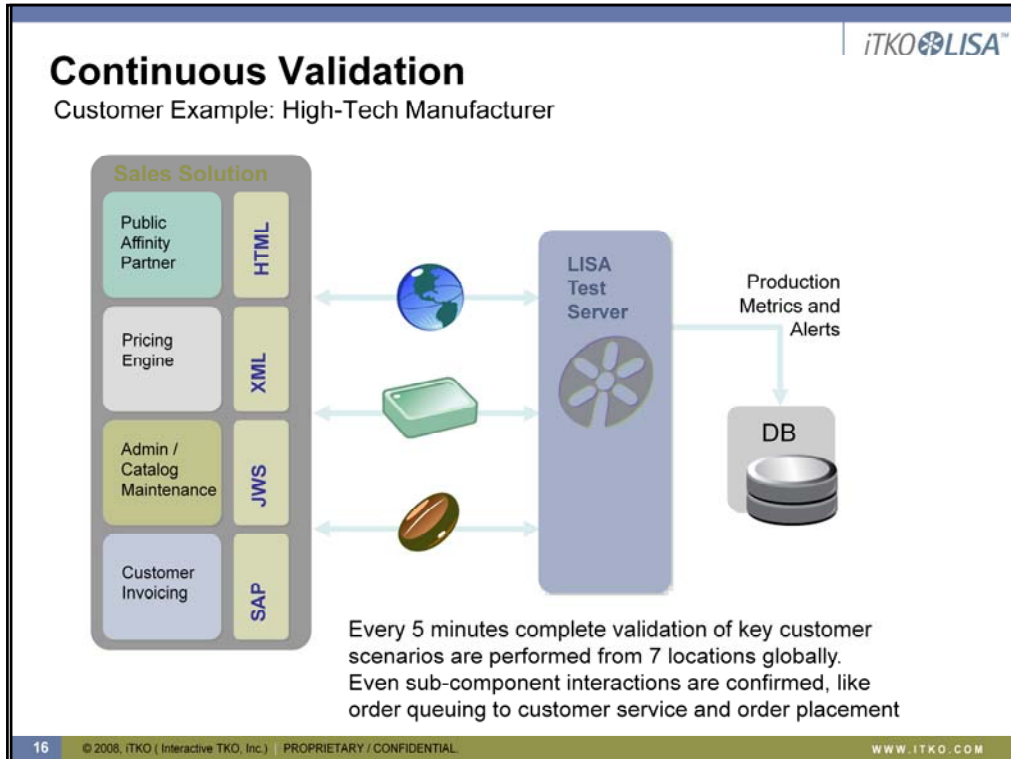
But OUCH, once App 2 hits production App1 is now failing. After taking way too long to realize, the App 1 team realizes that a data service is no longer working as expected. So guess who's upset? But wait, the data service wasn't even changed! So it turns out that an internal service used by the data service is now subtly different. That difference is causing an upstream failure.



Our goal here is to take some of the concepts your developers may have embraced around Continuous Integration and continue them throughout the lifecycle. The very reason you need it at build time is why you need it at the point of staging, in your integration labs, in partner development labs, and even in production.

To support this LISA provides build-time support for Junit and Ant natively. This way we hook into the build-time continuous process.

But there is no existing mechanism anywhere else, so for integration and production we provide our own infrastructure we call the Continuous Validation Service



A leading high-tech firm planned a very ambitious project to improve the usability and performance of its market-facing website within a 6 month time frame. It was quickly determined that running a small suite of use case tests, and manual testing, wouldn't provide enough test coverage of structural, functional and performance attributes to ensure that the multi-tier system could handle 1000+ concurrent users, as well as support

Using LISA, the development and testing teams created a suite of more than 1200 tests that are run automatically every night, and within 4 weeks the team has achieved more than 95% coverage of their application.

In addition the team has LISA continuously validating key customer scenarios every 5 minutes to ensure that as systems in the US, China, India and Europe are continuing to handle user traffic and transactions accurately and within Service Level Agreements.

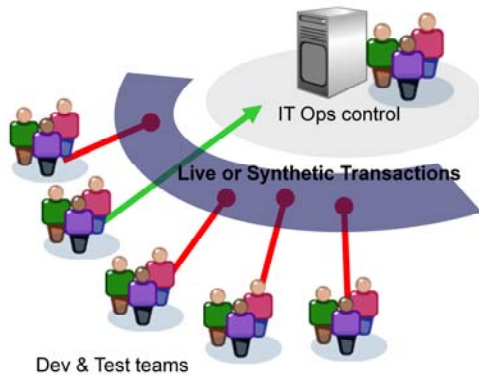
All of the continuous test data and alerts are then fed from LISA server into the company's IT Operations solution database for continuous monitoring.

As of now, the system has had no serious downtime problems in the last year, thanks to having such a strong set of regression testing and early warnings of performance issues.

Constraint Virtualization

Removing dependencies in the SOA lifecycle

iTKO LISA™



Teams need loose coupling too!

- Dependencies in the deployed system manifest as significant team-to-team constraint issues.
- Teams compete for shared platforms and unavailable, incomplete components, blocking agile and parallel development.
- Data volatility and inconsistency constrains development, quality, and performance validation.

17

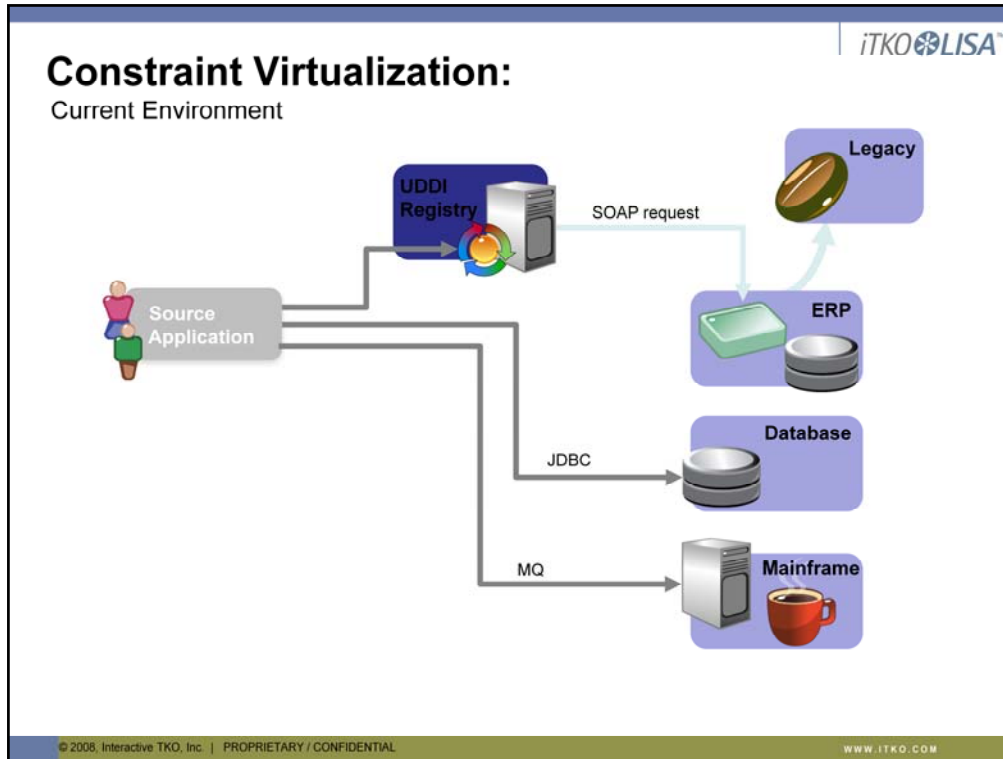
© 2008, iTKO (Interactive TKO, Inc.) PROPRIETARY / CONFIDENTIAL

WWW.ITKO.COM

Our 4th C is maybe the one most responsible for agility-robbing friction in development cycles.

(read the bullets word for word)

Constraint virtualization is essentially an acknowledgement that we must eliminate the constraints present in our dependencies. I'll use an example to illustrate...

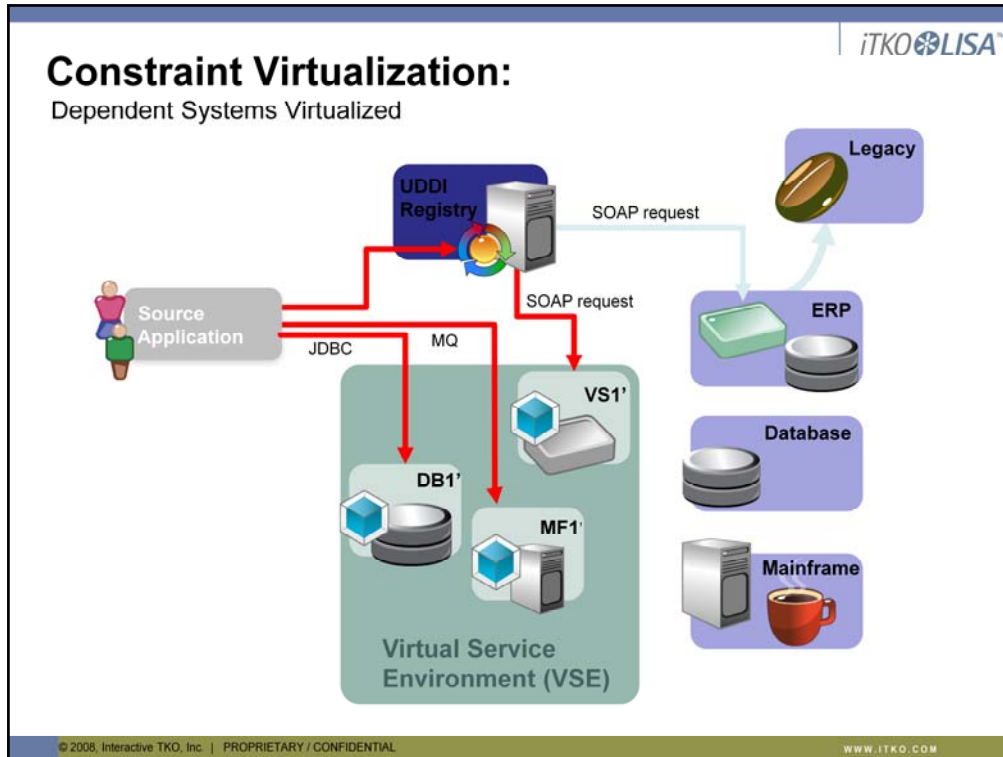


We are building or changing this “source application”, and we call out to 3 different systems. From the bottom up, the MF requires no changes to support our application, but we have restricted access to it. I’ve heard from customers that they only get 2 hours a day of access for example. So how do we develop our application with such constrained access?

The database doesn’t belong to us, and we need a few new stored procedures for our application to function. So great, I now have a months-long negotiation and development cycle to wait until that database is ready for my use. I’m dead in the water until then. How can I work now when a dependent system isn’t ready for me?

And finally, the web service I will invoke is still under development. Some operations work, and some don’t. The best part is it’s a day-by-day prognosis! Some days I hear that “hey the ERP guys latest build is broken” and my team is suddenly stopped until the other team clears the error.

I have a scope and a date to deliver to, but how can I with these constraints?



The answer is to virtualize the behavior of these systems. I don't mean server/hardware virtualization. That's great and use it. But you can't create a VMWare image of a mainframe. You can't stabilize a web service by imaging it.

But you can image the behavior of these systems, or model their future behavior.

Take that mainframe you don't have access to, image it's behavior, and you can host that image in a virtual environment that allows 24/7 scalable access.

Image that database, then modify the image, not the real DB, so you have the future state of the database. This allows you to start work immediately.

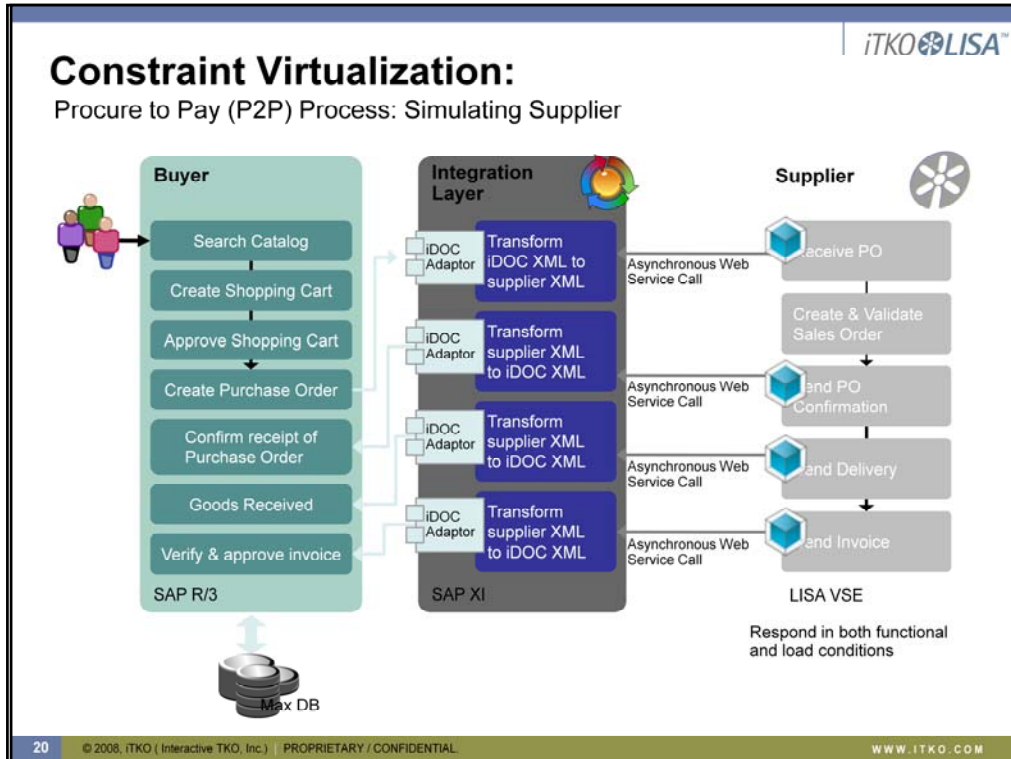
And for the web service, you can access either the live service or our virtual service as needed. As we show above, you can even do so from UDDI queries to bind you to the desired service endpoint dynamically.

Now that you have a strategy for eliminating these dependencies you are in control of your delivery destiny.

Our product that performs this magic is called LISA Virtualize. It contains a server called the virtualize service environment, or VSE, that is used to host your service images. This maps exactly to the server virtualization you do with VMWare or Citrix.

In those products you image a whole hard disk, then deploy that image into a virtual environment that knows how to run them. We do precisely the same type of activity, only at the service behavior level.

In cooperation, you have a completely virtualized, de-coupled development, test, performance testing environment.



Procure to Pay: In this test scenario, LISA is acting as a supplier. A purchase Order is created in R/3, which is configured to send a message to XI on order creation using iDOC adaptor. SAP R/3 Native iDOC is converted to iDOC XML, which is then mapped into Supplier XML and sent to the supplier as a one-way web-service call.












In the scenario, LISA is acting as a supplier. LISA gets the asynchronous message (using LISA VSE), parses the data and sends N number of PO confirmations based on number of items.


Here LISA is invoking one-way WS into XI. When XI gets PO-Confirmation, it maps it into outgoing iDOC XML and then hand it over to iDOC adaptor, so that the confirmation can be delivered to the SAP R/3.

iTKO LISA™

Our Solution: iTKO LISA

Increasing Agility & Quality for Complex, Changing IT Environments

| LISA Test | LISA Validate | LISA Virtualize |
|--|--|--|
| <ul style="list-style-type: none">  Functional Testing  Agile Regression Testing  Load & Performance Testing <p>LISA Extensibility Kit</p> | <ul style="list-style-type: none">  Workflow Pathfinder  Continuous Monitoring  SOA Policy Validation  IT Operations Metrics | <ul style="list-style-type: none">  Service Simulation  Behavioral Modeling  Performance Environment  Virtual Service Management |



LISA Foundation

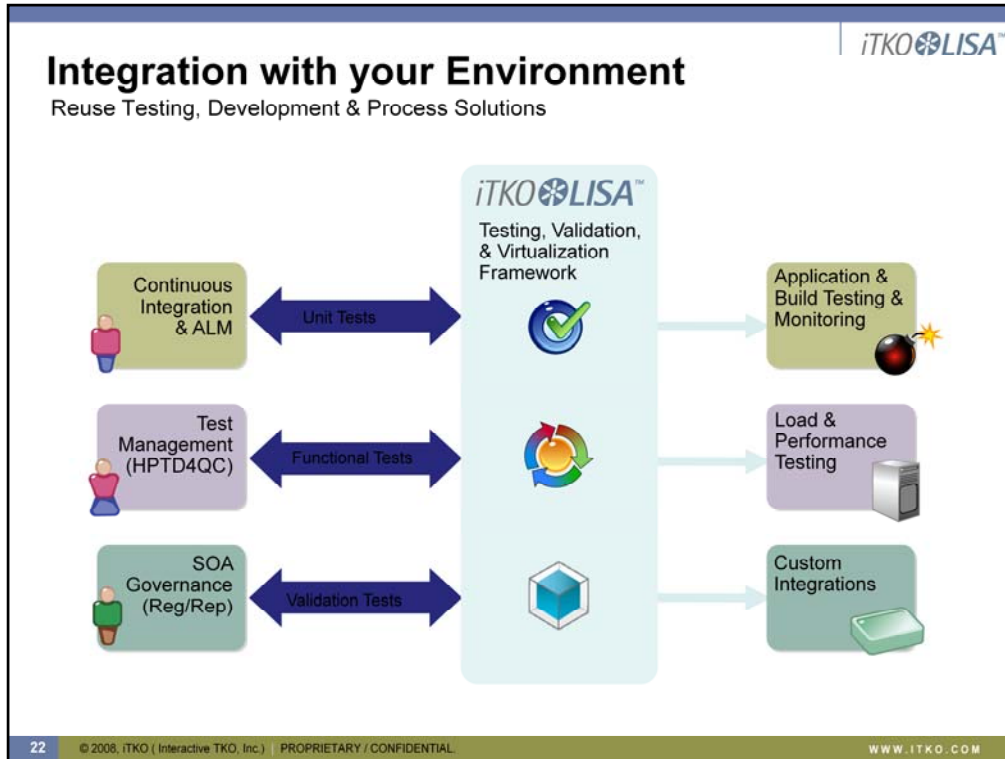
21
© 2008, iTKO (Interactive TKO, Inc.) PROPRIETARY / CONFIDENTIAL
WWW.ITKO.COM

So those are the 4-C's. Again we are convinced that good strategy for quality & agility in SOA is based on these principles.

And as I've discussed along the way, LISA is a great way to provide the technology support for this strategy.

(describe each product briefly)

An important point to make here finally is that this is actually one framework that is delivered in 3 personalities. It's one download and dbl-click install. You can get one license of LISA that does everything on this slide. Or you can license just our virtualization or continuous validation capabilities, for example.



OPTIONAL

LISA will play nicely with others. Most of the time that's HP Test Director for QC and LoadRunner. Sometimes it's also Junit tests. Either way, we are not here to rip-and-replace what's already working. We will fit into what you are already doing well and make you better at these new areas of concern.

We are cooperative with other tools in most of our accounts. It's only those who are not succeeding with existing tools or have no tools that adopt LISA for every facet.

Integrate with HP/Mercury's Test Director for Quality Center
 Bi-directional link to load test info from LISA, execute from TD for QC
 Introduce TD to an Agile/CI Process, Also supports leading ALM tools

iTKO LISA: Our Ecosystem

iTKO LISA™

Technology Partners | Delivery Partners | Channel Partners

The image displays a grid of logos for iTKO LISA ecosystem partners, categorized into three main groups: Technology Partners, Delivery Partners, and Channel Partners. The logos are arranged in a grid format, with each logo representing a different partner. The Technology Partners include SAP, Sun, software.com, webMethods, TIBCO, bea, ORACLE, hp, i2, vmware, SOA software, MKS, RALLY, dynaTrace, ca, wily, and Skytap. The Delivery Partners include CSC, accenture, P, iipt, Cognizant, patni, Infosys, Satyam, TATA, WIPRO, Adepta, Kvaaga, LOGIMETHODS, pureIntegration, INTELLIGROUP, and sgt. The Channel Partners include astute IT, ArchitectGroup, ionidea, IEI, Penta.T, POINTSOFT, and Systemation.

23 © 2008, iTKO (Interactive TKO, Inc.) PROPRIETARY / CONFIDENTIAL WWW.ITKO.COM

And finally it's important for you as a company to know that we are working with the same partners that you have partnered with. Many of our technology partners are actually customers first. That shows you how strong LISA is and how much we are committed to those platforms.

Many of you will have one or many SI partners. We are strategic partners with most of them.

Summary

- **Inadequate Testing:** iTKO LISA provides out of the box solution for business process validation and heterogeneous SOA applications
- **Unintended Consequences:** Validation across the SDLC from development to integration, through production, with one tool integrated into existing infrastructure.
- **Dependency on Constrained Systems:** Removal of team constraints on key systems, increasing agility of SOA & integration processes.



iTKO LISA™



For more info on LISA's SOA Testing, Validation & Virtualization capabilities and methodology:

- URL: <http://www.itko.com>
- Request an evaluation: sales@itko.com
- iTKO Blog: <http://blog.itko.com>
- Email: info@itko.com

So to wrap up, I introduced this whole notion around complexity taking away our quality and/or agility. I mentioned 3 specific areas of concern that are shown here. Then I introduced you to our 4-C's strategy to solve for these issues.

LISA is uniquely capable of delivering on the 4-C's.

(OK guys, it's up to you now ;)