Application Instrumentation, Stress Testing and Production Management:

What Software-plus-Services ISVs Need to Know

Microsoft Software-plus-Services

Thursday, 4 June 2009

Version 7.08

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Microsoft grants you the right to reproduce this white paper, in whole or in part, specifically and solely for the purpose of understanding licensing implications for hosting scenarios using virtualization.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

☐2008 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Hyper-V, SharePoint, SQL Server, Windows, the Windows logo, Windows PowerShell, Windows Server, and Windows Vista are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

CONTENTS

LIST OF ACRONYMS

Acronym	*Meaning*
APM	Application Performance Management
KPI	Key Performance Indicator
OS	Operating System
SaaS	Software as a Service
SLA	Service Level Agreement
WMI	Windows Management Instrumentation

INTRODUCTION

Developing Software-plus-Services applications is not a trivial undertaking. Decisions need to be made regarding multi-tenancy, scalability, the development process, go-to-market strategy, etc., in order to enable a smooth launch and ongoing delivery of the application. This paper will provide a successful and highly efficient process and describe an associated technology to enable the launch of any new Software-plus-Services application with a high degree of confidence.

This white paper will explore the following topics:

- Application and end-user experience instrumentation;
- Pre-Production and Ongoing Stress Testing – how it should fit into the overall application development and deployment processes;
- Production Application Lifecycle Management – how to leverage people, processes, and instrumentation to lower application delivery costs and improve the quality of delivery.

A key element in the successful launch of a Software-plus-Services application is the introduction of a Application Performance Management technology. Inclusion of Application Performance Management (APM) technology and services is critical to the successful deployment of Software-plus-Services applications for a number of important business reasons.

1. Customers need to trust the ability of the application to deliver the business value they've signed up for. A best practices approach to address this challenge is to provide near real-time customer facing dashboards showing the availability and performance of the application being delivered. Substantial value is delivered to the end customer via dashboards which include business relevant metrics based on usage of the platform. These metrics can include values for opportunities generated, new revenue acquired, cost savings, conversion rates, etc.
2. Contribute to prospect confidence during the selling process. Giving the prospective customer general dashboards establishes the business relevant benefits from APM. This approach is simple and highly effective.
3. Improve bottom line results by reducing operational expenses. Simply having a unified view of internal and external application performance and providing proactive issue notification to the internal or external technology operations team enables and promotes efficiency. These efficiency gains provide a means to achieve critical mass quickly, improving the bottom line, and providing the pricing flexibility needed to win new customers.
4. Increase revenue by identifying areas where you can improve application efficiency. All software systems have opportunities for improvement. Combining the in depth detail of APM data with customer click-stream data can create powerful business analytics which can help improve user conversion rates, reduce customer churn, and generate additional customer up sales.

This paper addresses the opportunity to substantially improve business efficiency through a systematic approach of application instrumentation, stress testing, proactive production management, and business intelligence integrated reporting for Software-plus-Services solutions. A Software-plus-Services application can be a SaaS application utilizing external services (i.e. Azure) or specialized client side software like a mobile client.

WHAT IS A SOFTWARE-PLUS-SERVICES APPLICATION?

Software-plus-Services should not be confused as another name for SaaS. A SaaS (Software as a Service) application is simply an application delivered as a service; it does not specify what type of application architecture is involved. A Software-plus-Services application combines software that is running in multiple locations together in a unified value proposition for users. A Software-plus-Services application may be sold and delivered as a product, as a service, or as a combination of both.

Examples of Software-plus-Services applications include:

- Applications that run on-premise but integrate services delivered from the cloud such as a trading application which pulls real-time data feeds from the stock exchanges;
- Applications that run in a hosted environment over the internet, but integrate back into the IT environments of their customers, such as hosted CRM systems that integrate into their customers' on-premise Active Directory or sales systems;
- Applications that integrate a user experience into clients outside the web browser, such as mobile devices, the Windows desktop or Microsoft Office;
- Combinations of the above.

Regardless of the architecture of your Software-plus-Services application, to deploy and manage it effectively you must be able to monitor its performance. This means you must have the ability to know how any given component of the application is performing at any time, regardless of whether that component is deployed on a customer desktop, on a customer server, on a hosted server, or in a remotely supplied web service. This white paper will provide you with guidance on how to effectively implement performance monitoring for your Software-plus-Services applications.

The System Shepherd® platform referenced in this paper is a perfect example of a complex Software-plus-Services application.

- Client side agent technology is deployed within your internal or externally hosted application environment.
- The agent technology sends application performance data to the externally hosted System Shepherd® backend environment via secure web services calls.
- Technical dashboards and alerts are delivered in a typical SaaS model via web browser and are accessible through mobile devices.
- Executive business performance dashboards for your applications are created via client side software from Business Objects. These are accessed either directly from the desktop software or via Adobe browser plug-in from a secure cloud report repository which then accesses real-time data via web services calls to the System Shepherd® platform.

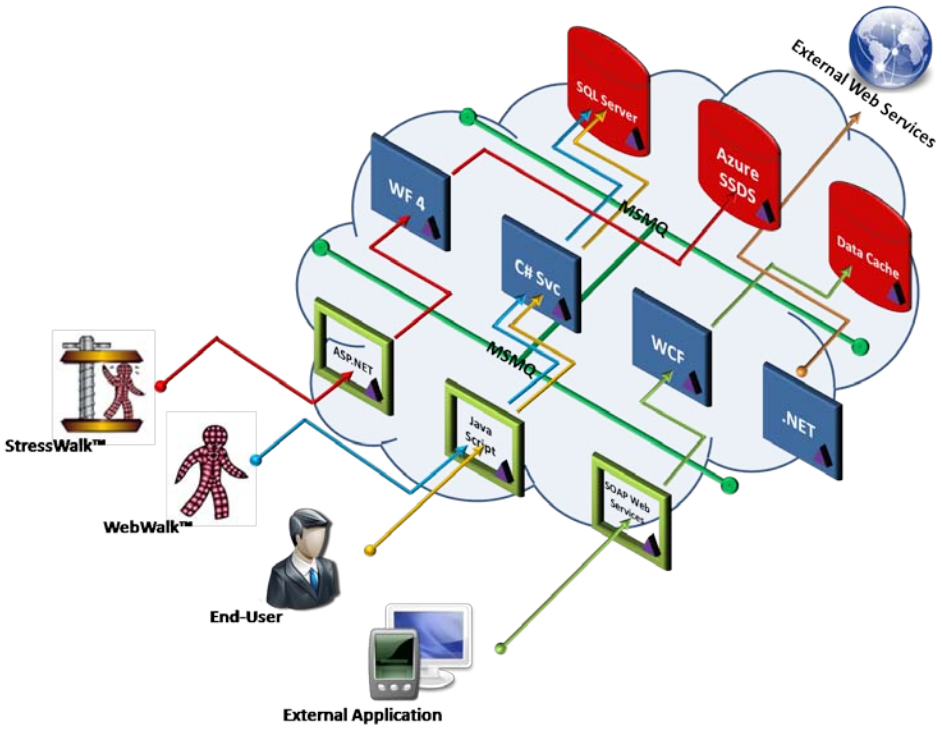
APPLICATION INSTRUMENTATION

Application instrumentation is the means to monitor and report the performance of an application. This instrumentation is done by using a combination of internal and external observation methods. Internal counters can be provided by the underlying operating system and by the application layer itself. External counters are typically used to measure end-user experience and other external environmental metrics. It is critical to combine internal and external measures into a single repository to maximize the business value of monitoring

Very rarely do organizations look at the operational and business needs during the initial phases of application instrumentation instead they look primarily at the needs of the software engineering team. This is a highly inefficient approach. Based on experience with a number of applications at various stages of development and deployment, a best practices approach has been developed that satisfies all three sets of requirements. A standard method for applying instrumentation allows engineering and operations to "speak" the same support language. This approach allows these groups to work in tandem to proactively resolve service problems much faster and provide better value to the business.

At a high level, the approach is simple. The groups standardize on the same instrumentation technology. Using a flexible, cost effective SaaS-based APM monitoring platform like Absolute Performance's System Shepherd®, the development staff can quickly deploy monitoring and performance visualization across their pre-production environments. An inheritance-based configuration scheme should be employed to enable seamless migration of monitoring configurations from the pre-production environments into the production environment. Likewise, meaningful performance measurements collected in the production environment are immediately familiar to software engineering personnel for side-by-side comparison with their development/test environment.

The areas of critical application instrumentation are illustrated in the following diagram.



Backend Processing Component Instrumentation

For most applications created in the Software-plus-Services paradigm, agent-based monitoring with very low resource consumption is the most reliable and extensible approach available to enable efficient instrumentation. Further, the most efficient instrumentation approach is to use a scalable template-based architecture such as used by System Shepherd®. This method allows for the creation of a class of infrastructure monitoring templates, which may be reused many times over. This methodology provides the ability to capture "one-off" metrics on specific devices while allowing configuration inheritance from the master template to take care of the rest. This will save significant time and resources allowing for a seamless and consistent migration from pre-production environments into production.

When looking at designing application backend components, it is important to take instrumentation into account from the onset of development. A lot of valuable data is available in standard monitoring configuration templates for most products via WMI or PERFLIB including MSMQ and .NET application component stack metrics. Many of the metrics exposed by WMI are counters which can be useful in their raw format for troubleshooting longevity related problems. However, they are often most valuable when converted to rate based metrics which should be inherent in the monitoring solution.

Many of the most important metrics are not available through standard WMI counters. Some application developers choose to create custom WMI counters. The article *"Powerful Instrumentation Options in .NET Let You Build Manageable Apps with Confidence"* located at http://msdn.microsoft.com/en-us/magazine/dvdarchive/cc300488.aspx serves as an excellent starting point for creating application specific counters. These counters can then be monitored via custom templates and thresholds within the monitoring solution. An alternate approach, which some development teams find easier to implement, is to log performance metrics and application events to a file. The monitoring solution should also be able to watch these files on a continuous basis and translate the metrics and events into rates using frequency based log analysis.

What custom parameters of an application should be monitored?

- Transaction details by user or client, or a combination of the two, can provide valuable insight into application utilization patterns by client or user within the application. These metrics can also be combined with resource utilization metrics to create a view of resource utilization by client or user. This information can then be translated into per-client or per-user delivery cost. This provides valuable pricing and margin data to the sales organization and enables the creation of per-deal profit and loss statements.
- Gathering internal application component or method call frequency as well as processing time is extremely valuable in determining where to begin the software optimization process. This will also point out where optimization will have the greatest impact.
- Monitoring external service call frequency and call duration (either internal or external) to web services such as Azure is critical. If there are performance or availability problems outside of the application, the development team needs visibility into where they are located so the team may either code around the problems or work with the external provider to resolve the issues.

Database and Data Access Layer Instrumentation

The data access layer is most likely composed of a SQL Server database and one or more in-memory caching services, and external web services calls. Basic database monitoring is fairly straightforward and handled by most products "out of the box". However, when building an application with instrumentation in mind, it is important to consider what additional information the monitoring platform should capture from either the database or external data service calls. It is recommended the following incremental metrics be used:

- Business relevant metrics such as revenue, user conversion, transaction volume, etc. These metrics enable the use of APM data for business analytics. This information is key to supporting continuous performance improvement at the business level.
- Resource utilization by user or client enables costing at the customer or user level as previously discussed.
- Aggregate application usage metrics help to facilitate both long-term capacity planning and feature utilization pattern analysis.
- External service call frequency and duration for data access by user or client provides valuable insight into the delivery characteristics of third party data and services providers.
- Data cache utilization and performance provides the visibility necessary to determine if the cache client software is effectively utilizing the data cache in place.

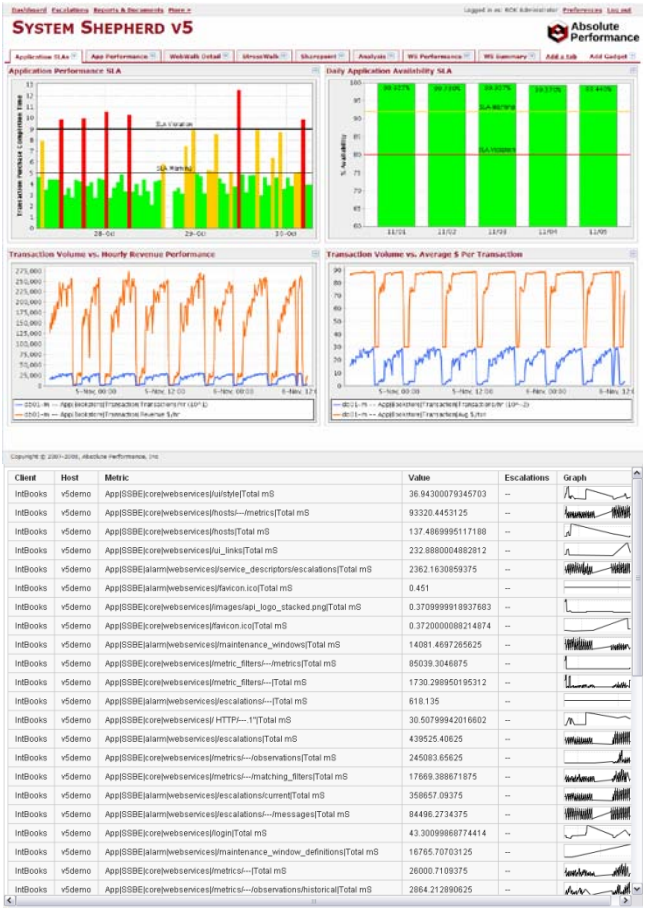
Some of this data can be gathered through custom queries to databases by the monitoring system while others require development team resources to collect metrics through the mechanisms identified in the previous section.

Front End User Interface and Web Services

Some of the most valuable business relevant metrics reside in the front end components of an application served by IIS or other web server components. As with the other layers of the application stack, the standard metrics available through WMI provide a good starting point. However, these metrics do not give visibility into the end-user experience. The recommended best practice is to combine this information with synthetic user transaction monitoring via a user transaction simulation service such as WebWalk™. This layer of monitoring allows for proactive tracking of the end-user experience. Alert thresholds should be set for metrics such as content/functionality validation, transaction execution times, and rates of change for transaction execution times. This component of external monitoring should be replicated behind the load balancer within the application environment to provide visibility into the behavior originating from individual servers or virtual machines. Combining external synthetic transaction monitoring with internal synthetic transaction monitoring enables the operations/delivery organization or outsourced provider to be proactive in managing the application.

In this tier, the value of the data appearing in access logs should also be monitored. This can be achieved by customizing the logging parameters to expose the user or client associated with each request and the amount of time it took to return the request to the browser. Frequency based log analysis can then be employed to capture access frequency and average/max/min response time metrics associated with the requests on an

aggregate basis. This data is highly valuable for isolating application response time issues. Additionally, the raw data can be used as part of the overall data set to identify resource utilization by an individual user or client.



Network and Storage Infrastructure

A holistic view of the application delivery infrastructure is not complete without looking at the performance and availability of the network and storage infrastructure. Access to this data is dependent upon the hosting strategy being employed. Administrator level access to the network and storage infrastructure provides a wealth of data. This data may be collected via SNMP with System Shepherd® agents, or just about any other monitoring solution. If access to the network and storage infrastructure is limited, data collection will require a bit of creativity. One method is to correlate data from ping and port monitoring from various points within the architecture with individual server or virtual machine networking statistics. This provides a view of network behavior. On the storage side, individual server (or virtual machine) physical storage metrics can be used to obtain a partial picture of storage performance.

Real-Time Application Analysis

Once the above recommendations are implemented, data will then be available for analysis. The key is to be able to use the data to improve application delivery from the perspective of the end-user and the business. Regardless of the tools or services used to collect the data, it needs to be accessible in one place to maximize the value of the monitoring solution.

The best practice in this area is to provide the technical teams with proactive alerting. Proactive alerting requires configurable alarm and escalation rules and the ability to perform real-time or near real-time analysis on the data in both ad-hoc and standard views. To truly provide proactive management of the application the following graphical analysis tools are recommended:

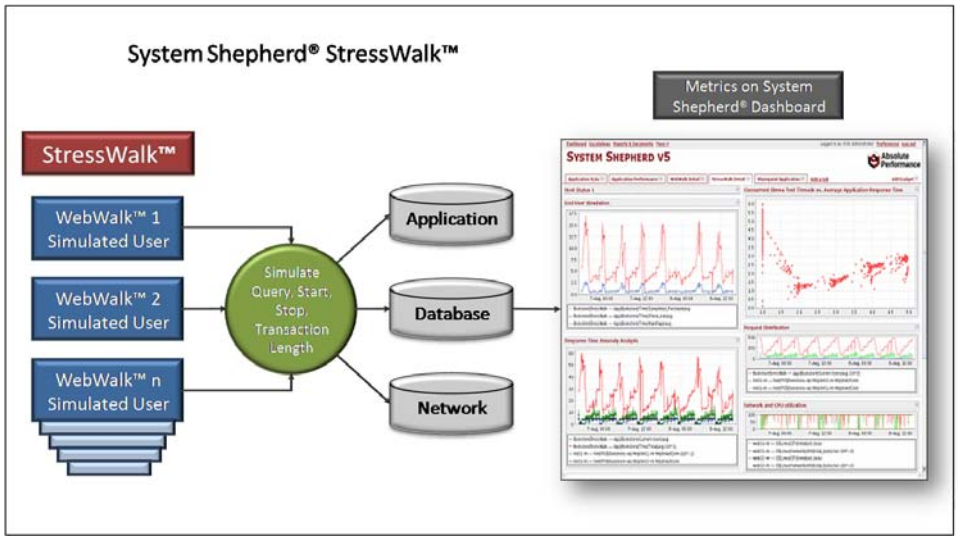
- Metric cross correlation with scaling – allows for the graphical correlation of any metric from any measurement point to be correlated with any other metric across the entire application architecture;
- Metric scatter plots – allows for identification of any trends that may not be evident in a time based graph;

- Metric watch lists with trend indicators – enables the operations team to analyze various metrics in a compact view with spark lines indicating 24 hour trends which can dramatically reduce root cause analysis time;
- SLA graphs – provides the ability to view at-a-glance whether or not the application or its sub-components are meeting internal and external business requirements;
- Custom tabs or workspaces – enables the creation of job-specific views of the infrastructure and are extremely valuable in troubleshooting recurring issues or difficult to "catch" problems;
- Web services data access – through standards compliant web services (SOAP 1.1/WSDL or Restful JSON) custom views can be easily created and reported via industry standard tools such as Crystal Reports™ and Xcelsius™.

PRE-PRODUCTION STRESS TESTING

When preparing for the launch of an application, it is crucial to understand how it will likely behave under real-world user load. Our experience indicates the best way to plan for growth and avoid surprises is to simulate the anticipated user load with a realistic and well-instrumented load test. Developing a test scenario for each of the application's anticipated user types ensures that all application functions are exercised. Furthermore, by simultaneously loading the application with multiple user types, we can expose complex interactions inside the application such as deadlocks, race conditions, and queue backlogs. Real-world application usage is rarely characterized by searching for the same keyword or logging in as a single user thousands of times. It is extremely important to introduce unique, dynamic user input to the load testing to avoid falsely positive results due to caching of sessions, query results, etc.

Effective load testing, like ongoing operational monitoring, should correlate end-user experience with internal application performance metrics. The load testing then serves as both a direct measurement of performance as well as a means to exercise the application while observing its behavior directly though database, application, and OS metrics. The second question after "What is the capacity of the application?" is always "Where did the bottleneck occur?" With a properly instrumented environment and well-designed load test, we are able to answer both questions.



Stress testing should not be a standalone effort. In an ideal world, it will take advantage of all of the instrumentation previously created and leverage the existing synthetic user scenarios. With System Shepherd®, WebWalk™, and StressWalk™ all of the components tie together to provide maximum reusability. Additionally, data from the stress testing runs should be available for analysis side-by-side with live production data thus enabling correlation and continued enhancement of the stress testing approach.

Testing Types

The chosen stress testing service approach should be tailored to specific business and testing objectives. In some cases, stress testing is part of an ongoing code release process, validating functionality and performance prior to pushing each new revision to production. For others, it is performance validation of a new platform, application, or hardware deployment. Using a stress testing service can be much more cost effective than purchasing a product with a permanent $20K+ per month cost you can expect to spend for the product + product support + load generation and data analysis infrastructure + training + a person. Results are produced

much faster and provide deeper analysis because the stress testing and application/infrastructure instrumentation are tied together in a single solution.

Failure Test

Execution of an application failure test is usually the first component of a stress test. User load is ramped up proportionally across usage scenarios until the application fails or experiences intermittent failures beyond the business defined thresholds. The critical analysis points for this type of test include:

- Failure mode identification. Are there one or more failure types (i.e. page timeouts, explicit application errors, and unacceptable response times)? Which application components experienced failures?
- Root cause analysis. For the software components which experienced failures, why? Was it caused by database deadlock conditions, a saturated connection pool, external service calls, firewall configuration, disk I/O bottlenecks, failure to use data caching components as designed, inefficient code, etc.?
- Relative expense or overhead of each application user type or scenario (i.e. content contributor vs. read-only user). Based on business assumptions, this will help determine prioritization of software optimization and new features.

Sustained Load Test

A sustained load test is designed to identify the longer term reliability and performance of the application. It is executed at 80% of the failure test capacity for several days or longer according to business requirements and risk tolerance. Analysis of any outright failures during this test is identical to the failure test analysis. However, special emphasis should be placed on analysis of gradual performance degradation over time. This type of test often reveals both software improvement opportunities as well as operational procedure improvement or automation opportunities. Critical analysis points include:

- Component performance degradation over time,
- Component memory leaks,
- Garbage collection induced issues (these typically take the form of periodic slow response),
- Database backup and automated maintenance induced issues,
- Long-term backend processing performance (jobs, queues, archiving, etc.),
- Intermittent component failures.

Ongoing Approach (Agile Stress Testing)

Effective application lifecycle management should specify stress testing as a prerequisite to production release. Testing each new application version under load provides a rigorous shakedown to expose elusive failures or performance issues, greatly minimizing risk. With retention of thousands of users at stake, the confidence and peace-of-mind following a successful test is often just as valuable as identifying a failure. Progressive development teams have instituted regular load testing even earlier in the lifecycle to quantitatively evaluate technologies, optimizations, algorithms, etc., based on real-world scenarios. Critical analysis points include:

- Validation and quantification of performance improvements,

- Validation of application functionality and stability,
- Benchmark validation against test results from the previous release.

Special Investigations

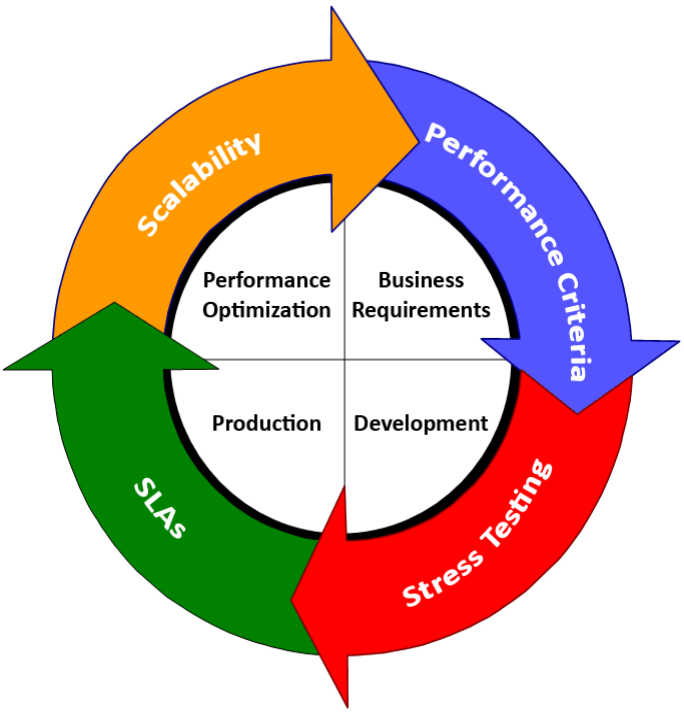
It is not a coincidence that the most difficult application problems only occur in production. Typically, complex real-world user activity trips the most elusive flaws in an application. A specialized stress test, designed (or modified) to reproduce a user-reported error can expedite fault resolution. Accelerating the incidence of application failures to a matter of hours – especially when combined with custom monitoring triggers to capture a snapshot of the application at the moment of an intermittent failure, can save man-days of expensive development resources. Utilizing stress testing in this manner can dramatically cut problem resolution time and reduce the longevity of a critical problem identified in the production environment.

Leveraging Pre-Production Testing for Production Visibility

When you execute pre-production stress testing, you should be able to leverage the work product to continuously enhance production user experience monitoring. Your chosen solution should use interchangeable configurations between production user experience monitoring and stress testing. The System Shepherd® WebWalk™ service (discussed in the Application Instrumentation section) makes use of the same user scenario configurations as StressWalk™ which ultimately improves the efficiency and effectiveness of both components while providing a significant cost savings to your business.

PRODUCTION APPLICATION LIFECYCLE MANAGEMENT – TYING IT ALL TOGETHER

Building and deploying complex business applications is difficult. It is insufficient to release applications or application updates to an ill prepared IT operations organization. To maximize the value and investment in new technology, a more holistic approach must be undertaken. The four key components to this approach are:



- Business requirements development with an emphasis on translation to operational requirements,
- Application development support process including stress testing and instrumentation which were covered in detail previously,
- 24x7 production application support inclusive of key ITIL based processes
- Continuous performance optimization which focuses on delivering more value to both end customers and the business leveraging the APM solution in place.

Business Requirements Development

The business requirements for an application typically evolve over time as the business changes and grows. Because of this, the business requirement development process needs to continue throughout the life of the application. Unfortunately, many companies stop this process in the development phase and are then ill prepared to make changes when the application is in production. The process for developing meaningful business requirements must take into account the many facets of the business. At the top level are market assumptions which relate directly to sales and marketing projections. These, in turn, translate to a multi-dimensional set of requirements related to the service being provided. During development, it is important to remember that business requirements are based on sales and marketing projections and to not allow the operational delivery requirements get too far ahead of actual sales execution. This approach focuses on basing the requirements on business projections rather than engineering capabilities.

As high level business requirements are translated into operational delivery requirements, accessible delivery metrics such as availability, end-user performance of the XYZ process, average screen response time, average operational cost per user, etc. should begin to be collected. Additionally, the requirements should dictate what type of support is needed: type of data center, outsourced 24x7 operations or internal delivery resources, an active disaster recovery site or a less robust DR plan, dedicated infrastructure or shared infrastructure, etc. Operational requirements are defined as those functions that allow the application to continually deliver the defined business service in a consistent manner.

24x7 Production Application Support

Ongoing development and improvement of the instrumentation that started in the development phase further ensures the ability to proactively deliver the application with the highest quality possible. Feedback from the production management team to development is critical. Enhancements to application monitoring happen here and are often not passed back to the development teams — ensure this will not happen.

Critical Application Management Processes

Best practices dictates that the ability to consistently and reliability deliver an application to the market can be achieved by utilizing an integrated and well defined set of operational processes such as provided via the ITIL framework. At the onset, this can seem to be a daunting and overwhelming challenge and expense. One approach is to adapt a simplified ITIL framework that first targets key process areas. We recommend a service strategy consisting of three main components:

- Service Management – supports the delivery and measurement of customer service;
- Configuration Management – the maintenance of IT asset inventory to appropriately supported configuration levels as well as the forward looking capacity levels sufficient to support evolving service requirements; and
- Service Operation – the core processes supporting the delivery of your application.

A properly configured APM solution whether purchased or delivered as a service should provide near real-time incident (alerting, notification and issues escalation), problem (data identifying root cause), change (supporting detail regarding the effectiveness of the change provided through monitored metrics and unauthorized change typically results in issue notifications), and availability (SLA reporting) management.

Change management processes are some of the easiest to implement and can have the highest payback in terms of increased application availability. Unfortunately, many companies fail to follow their change processes because technical staff members find the processes to be inconvenient and cumbersome. For most of our customers we advocate a very simple change process consisting of documentation, signoff, and defined maintenance windows with the flexibility to accommodate emergency changes. Whoever is responsible for implementing changes should be held directly accountable for meeting the availability SLAs of the application. Properly implemented application monitoring should provide data for analysis which supports the need to implement performance-based change to meet business service levels.

A well defined, documented and tracked change management process is probably the most important component to making sure your instrumentation strategy evolves to keep pace with changing business needs.

The ITIL support processes identified in the support model above are required to produce the service results demanded by enterprise customers of Software-plus-Services applications. If this type of structured approach is not in place, the investment of time necessary to implement these processes internally or switch to a service provider who does will provide rapid payback in terms of customer satisfaction due to higher quality service delivery, customer retention, and operational efficiency.

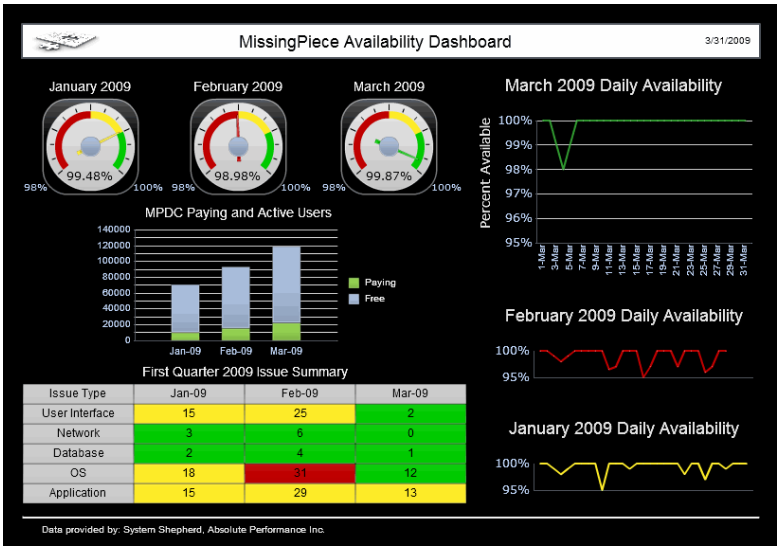
Application Monitoring Enables Proactive Efficient Management

Proactive application monitoring provides the insight needed to enable efficient production management. When implementing the monitoring elements discussed in this paper, production operations staff, whether internal or external, will now have the visibility necessary to manage the Software-plus-Services application at a lower cost and higher value to end customers. The next step is using this proactive intelligence as a selling tool by providing some level of transparency and quality assurance to customers and prospects alike.

SLA/KPI Visibility for Customers

Leading SaaS companies such as Salesforce.com provide SLA visibility to customers based on internal APM data around service availability. This provides assurance to enterprise customers that their businesses are not at risk using the Salesforce platform. Providing that same assurance and more can easily be achieved by implementing the approach outlined in this paper.

One option is to offer a read-only System Shepherd® dashboard to each customer. The top figure on page seven illustrates what this



can look like. Alternatively, a higher level custom executive dashboard can be created using the exposed web services data. The metrics exposed in this type of dashboard should be relevant to the end customers' business. Typically, the availability and aggregate performance of the solution in conjunction with two or three business relevant metrics or Key Performance Indicators (KPIs) relative to the customer's use of the platform are a good place to start. For example: transactions per day, feature utilization rates, and cost savings per relevant unit. With KPIs such as these in place, feedback can then be elicited from customers on how to make your application and the dashboard more meaningful to their business.

Continuous Performance Optimization

Once the Software-plus-Services solution is deployed, you want to continuously look for ways to:

- Reduce operational expenses on a per-unit basis,
- Improve end-user experience,
- Drive more revenue, and
- Empower the sales organization.

An outsourced application support vendor(s) should be employed to help deliver these objectives. With sound APM data and a flexible operations team the organization is now poised to produce these results to the business. The following approaches have been found to be effective:

- Empowering a technical operations vendor or internal team to communicate continuously with the software engineering team to address minor performance and scaling issues on a prioritized basis before they become customer impacting. Having consistent application and infrastructure data between the production and non-production environments accessible to both groups makes this process much more efficient and productive as the trust barriers are eliminated.
- Implementing a Monitoring 2.0 solution to extract business relevant data from the extensive APM data being collected. The general concept of Monitoring 2.0 is to integrate APM data with user click-stream, delivery cost, and per customer or user revenue data into a physical or virtual data warehouse and then use business intelligence techniques to answer questions such as:
    - If 10,000 users are added for a large prospective customer, how much incremental delivery expenses will be incurred? (the answer enables a large deal P&L approach to pricing)
    - As the application user base scales, what does the delivery expense curve look like?
    - What are the most frequent usage patterns within the application?
        - Which of these usage patterns lead to increased revenue?
        - Which of these usage patterns lead to customer churn (turn over)?
    - Which features within the application influence prospect to customer conversion?
    - What is the prioritized list of features within the application that should be enhanced or eliminated?
    - Which features or application components have the highest operational support expense impact?

SUMMARY

Implementing an end-to-end monitoring and management approach will have a significant impact on successful Software-plus-Services businesses. The top items covered in this paper were:

- Complete end-to-end application and end-user experience monitoring is essential to understanding the technology and can add great insight into the business. This instrumentation is the foundation for reducing operational expenses and improving customer experience.
- Stress testing should be integrated into the application release process. It illuminates application problems before customers see them and enables the release of new software with confidence.
- Exceptional application management requires a process and analytics oriented approach. The stronger the processes and underlying application visibility are, the more the end customer will love the application for its stability, performance, transparency of delivery, and continuous improvement.
- Monitoring 2.0 – the combination of business intelligence and application performance management –provides significant benefits to the business by answering important questions such as "How can we increase revenue through an increase in free to premium conversions?"

ABOUT ABSOLUTE PERFORMANCE

Absolute Performance enables business executives to deliver mission critical applications at lower cost and higher value to their customers. We do this through a combination of software (monitoring delivered as a service) and application management services. Our production application lifecycle management approach encompasses business requirements translation to operational delivery requirements, pre-production stress testing and application instrumentation, production application management services and continuous performance improvement via proactive capacity planning, continuous application performance optimization and Monitoring 2.0 business intelligence services. You can purchase our services directly or through one of our channel partners who can be found at http://www.absolute-performance.com/partners.

For more information:

Jerry Champlin, CEO

jerrychampin@absolute-performance.com

www.absolute-performance.com

303.565.4444