WHITE PAPER

Leveraging Performance Optimization for Business Advantage: Bridging the Life-Cycle Gap Between Development, Transactions, and Production

Sponsored by: dynaTrace software Inc.

Melinda-Carol Ballou
August 2008

IDC OPINION

One of the great challenges for IT organizations is managing the application life cycle from development to production — using information obtained from automated tools to pinpoint performance problems to find and resolve them throughout the software life cycle and, most importantly, when they occur in production. The costs of software failure for business-critical applications are visceral, with IT developers and operations staff often spending inordinate amounts of effort and time seeking the root cause of application performance issues. Typical development and monitoring tools help to identify the existence of problems but tend to give little detailed diagnostic visibility into single-tier and multitier transactions. Frequently, developers engage in tedious information-gathering approaches that result in palliative "fixes" — and potentially lengthy service outages. These tactical approaches may alleviate application performance issues initially but often treat only the symptoms and not the underlying problems. Given the business-critical nature of these applications, it is key to development, operations, and business stakeholders to evolve strategies to increase efficiency, cut expenses, and improve timing with regard to application performance and diagnostics. The costs of downtime for such applications are prohibitive and rise exponentially the later in the life cycle they occur and are discovered.

This White Paper considers the business, IT, and operations challenges of bridging this gap to enable users to proactively find and repair performance and transaction problems in distributed applications (from development to deployment).
SITUATION OVERVIEW

Disruptive Market Trends Driving Greater Application Life-Cycle Management Rigor

As business analysts, stakeholders, and IT and operations teams seek to collaborate about software requirements and deployments, they often struggle. This can occur even when commitment to collaboration exists and these groups use their best efforts to connect with one another. Organizations remain stymied by the cultural divide that exists across groups, poor communication channels, as well as process and organizational issues. (These issues include lack of availability and lack of engagement by key business stakeholders, architects, and operational and development staff.) As a result, IT projects too often are irrelevant to changing business needs, which can hamstring the business and impair competitive success. These issues are exacerbated by poor application performance and lack of visibility into the root cause of performance problems. Emerging, complex infrastructures with Web 2.0 and service-oriented architecture (SOA) add levels of complexity and opportunities for additional problems to creep in from a design and architectural perspective.

Lack of visibility into where and why application problems occur drives finger-pointing and guesswork, with developers often creating short-term "fixes" by adding hardware or virtual machines to increase processing power in an attempt to address performance issues. But as long as the actual causes for poor application performance remain unknown, these tactical solutions can create performance problems elsewhere, or they may remain a temporary "fix" until the next major transaction surge or other issue causes a repeat problem.

Even where communication across teams is strong, business change and lack of common visibility into individual transactions across groups can still drive misunderstanding and poor iterative development approaches as the project progresses to production. This can lead to inadequate quality approaches as well; it's challenging to provision quality software if it's unclear what transactions are doing across single and multiple instances in deployment. An obvious decline in quality and business responsiveness can result from lack of coordination between development, testing, performance monitoring, and analysis and diagnosis of transaction processing. Dysfunctional scenarios can and do result, such as development systems where teams have little or no insight into actual problems after spending weeks typically developing, testing, monitoring, and seeking to deploy the software. This can result in unacceptable application performance or even downtime losses of customer-facing, 24 x 7 applications that may be unavailable for significant time periods.

These challenges are occurring in the context of a market that is trending now toward more rigorous approaches to managing testing, monitoring, and other life-cycle phases (such as change management and project portfolio management). A variety of factors create a "perfect storm" of enablers and drivers for this evolution toward effective development, quality, and management practices.

These factors include complex sourcing that demands closer coordination across offshored, outsourced, and distributed internal IT staff. Compliance initiatives such as Sarbanes-Oxley provide both funding and the legislative imperative to adopt more rigorous approaches to managing the software life cycle and carry with them targeted
requirements for every business. While these two issues are core drivers, another is the emergence of SOA approaches to development, which necessitate closer business and IT collaboration for services to be useful and pragmatic.

Beyond these areas, a need to better manage security earlier in the life cycle brings security best practices to the forefront earlier in the life cycle. Where security in the past tended to be addressed at the testing phase and involved a separate, security-focused group, increasingly IDC sees Global 2000 organizations incorporating security best practices up front at the requirements phase, drawing in development teams in coordination with security staff (though positive collaboration between these two teams is emerging slowly).

These issues create an impetus for closer integration across life-cycle phases and stronger process and organizational approaches for improved quality and diagnostics for distributed applications. These factors also drive incentives and funding commitments from executive management as a result of the visceral impact of application life-cycle management failure on the business. This is exemplified in the confluence of software quality life-cycle management with monitoring and diagnostics.

The Push for Quality Coordination and Effective Code Analysis

Increasingly, IDC sees testing, monitoring, and emerging life-cycle diagnostic approaches coming together to improve the combined areas. Proactive, up-front diagnostic information about existing problems such as architectural validation improves code quality earlier in the application life cycle. The earlier that application performance problems are discovered and fixed, the less expensive it is to repair them. (IDC research indicates that fixing problems later in the life cycle increases costs by 10 to as much as 100 times, depending on how close to deployment and additional code dependencies related to the changes.) Further intangible costs such as damage to company reputation as a result of poor postdeployment user experience may be prohibitive competitively and can erode corporate brand equity. In an increasingly difficult economy (expected 2008–2010), margins for error will be thinner and less forgiving when business-critical software goes down or performs poorly.

There are multiple benefits to linking effective application diagnostics with automated testing and application performance monitoring. While those automated tools uncover problems, it is helpful to have context for how an application's individual transactions progress in distributed environments, to be able to assess where and why performance problems occur. In that context, intuitive diagnostic products that are usable by those who are nontechnical and not savvy about development, can help address the communication divide across groups. Whether someone on the operations team needs to identify the location of a problem to provide context for a developer or partner, or an end user on the business side needs a context to demonstrate his or her frustration with the application, ease of use and ease of reporting are key elements for success with diagnostic tools.

But what sorts of tools could work well to enable communication with regard to application diagnostics, in conjunction with monitoring and typical testing automation?
Framework for Emergence of Diagnostic Products

Key to analyzing, diagnosing, and fixing poor application performance are the ability to:

- Identify which specific transactions — in 24 x 7 production with 100% transaction coverage — are experiencing performance problems in an application
- Have visibility into the specific transaction’s execution path in the context of the surrounding environment in which it has been configured (e.g., server topology, application architecture)

A complex range of issues and problems can impact performance, obviously, from poor use of existing software frameworks to resource contention, ineffective remote access approaches, and errors resulting from poor design, poor coding, or poor configuration settings, to name typical causes. Analyzing the root cause of these performance issues requires discrete individual transaction paths. Traditional statistically based application performance monitoring solutions are typically incapable of providing this level of detail in production. Instead, they tend to give visibility into aggregate and average data without showing the outlying application issues. (Averages can hide the outliers that may be responsible for the issues.)

Along with the emergence of technologies and development paradigms that enable faster deployment, such as Web 2.0 and SOA, we see the emergence also of limited understanding and inappropriate expectations about the performance of those deployments. Given that distributed application environments dominate enterprise software deployments, it becomes all the more important for users across the board to have diagnostic insight about individual transactions in both single and distributed server environments. Given the organizational and process schisms across groups described earlier, these tools would ideally be able to be used by staff ranging from business analysts to developers, testers, systems analysts, and architects for operational deployment and provisioning.

The relationship between effective development, testing, production, and strong diagnostics is key for Global 2000 companies that rely on software to run their businesses. Yet, all too frequently, these approaches and automated technologies are poorly integrated.

We now consider the capabilities of emerging vendor dynaTrace software as an example in this context.
Solution Capabilities for "PurePath" Visibility to Facilitate Diagnostics

dynaTrace software was launched in 2005 and provides performance management tools that incorporate diagnostic capabilities. Headquartered in Linz, Austria, and Waltham, Massachusetts, with around 70 employees and 75 customers and funded by Bain Capital and Bay Partners, the company released its first solution during 1Q06.

dynaTrace's technology augments monitoring with application diagnostics. This helps to make specific transaction problems visible so that users can go beyond typical application monitoring to resolve performance problems in single-server and multiserver heterogeneous environments. When used earlier in the life cycle, dynaTrace's tools help users prevent issues proactively early in the software development process by supporting detailed architectural validation and performance regression testing as part of a continuous integration process. When used in production, dynaTrace's diagnostic capabilities have enabled some users to cut the time and expense of trying to find, reproduce, and fix application performance problems by as much as 30%, according to several case study references, two of which are summarized as part of this white paper. Time is of the essence when seeking to isolate and find problems and then address poor-performing, business-critical applications. Mean time to isolate and find problems creates even greater efficiencies of scale (see case studies).

dynaTrace's technology lets users identify specific application services that are affected by poor performance. The product traces an application's discrete transactions and also gives context to help users diagnose the root cause of application problems to more rapidly resolve issues:

- During execution of transactions, dynaTrace collects data to provide information about the transaction's path of execution through its PurePath technology.
- The data gathered includes specific information such as relevant sequence, timing, resource utilization, and data about the transaction's execution steps.
- dynaTrace makes the transaction path visible to users, whether the transaction is running across multiple logical or physical tiers, even in a heterogeneous .NET/Java environment.
- In an effort to help minimize performance overhead and increase accuracy, dynaTrace's agents gather and send data in real time to a centralized server for offline analysis. No local processing is performed on the monitored server.
Two key benefits that can accrue to users as a result of this approach are:

- Helping to proactively prevent performance problems by gaining insight into specific dynamic application behavior during development and deployment
- Helping to reduce the time it takes to fix unacceptable application performance by letting users reconstruct the context for specific, rogue transactions quickly (This, in turn, enables developers and/or suppliers to find and repair problems efficiently through the contextual data gathered by PurePath technology.)

Oftentimes development teams don't know what's going on under the hood of an application — they may be using a framework, for instance, to speed software creation, but they don't understand its inner workings. As virtualization becomes more popular, many applications may run in virtualized mode, adding to the complexity of distributed infrastructure for deployment.

Major user pain points include little or no visibility into the causes of poor application performance so that it takes much too long to isolate, find, and then resolve crucial performance problems. In addition, many customers are isolated and reactive since various departments — from business stakeholders to development and operations — work separately from one another, as was discussed earlier. In addition, varying groups use varying toolsets, which cause problems due to lack of consistent data access and common information.

dynaTrace can help to connect different departments — from users to development, quality assurance (QA), staging, and production — since it provides a common performance management and diagnostic solution across groups and development phases. Emerging integration and partnerships with other life-cycle products (HP Software, IBM, Borland, Eclipse, Microsoft Visual Studio) enable users to integrate information from development, load testing, and monitoring into a diagnostic session. dynaTrace and PurePath provide users with the context they need to reconstruct a problem in their local environment. Therefore, users don't have to reproduce the environment, since the entire transaction environment is reconstructed automatically.

dynaTrace solutions help to solve the visibility problem by monitoring specific, granular data points at the business transaction level. These solutions help identify which server, which user, what client, and which transaction is affected by poor performance. As slowdowns become visible early on, those responsible for analyzing performance have context and early insight into potential problems, how important they are, and what the impact may be. This transfers into the business benefit of quick resolution of problems that can otherwise remain impenetrable due to lack of causal information.

To fix application performance problems, developers need to know what happened, where, when, and why. They need to reproduce the issue for detailed diagnosis, which is often very complex and not possible due to lack of visibility and lack of context. In addition, since problems often become apparent at deployment time (that weren't discovered during development), production teams need to give developers as much information as possible to efficiently fix and resolve the problem. Production needs to document and triage application problems, yet those teams are often not well-versed in application development issues or code analysis. They need to
communicate the issues to developers or suppliers who will be responsible for fixing them. Without context, finger-pointing can be ongoing about whether a performance issue is related to system configuration or software problems. And visibility also helps developers visualize the performance problems of business users who are experiencing unacceptable slowdowns.

dynaTrace’s PurePath technology is able to trace each and every discrete transaction in the environment. dynaTrace measures CPU consumption and provides other contextual information for transactions. For instance, dynaTrace’s technology measures issues such as memory consumption and exceptions and record how many objects are transferred over a wire for SOA, database calls, and memory leaks, as some examples. For any measured transaction request, dynaTrace users get this kind of detailed information and are able to track that as part of a diagnostic session for a specific time frame. Users can then send diagnostic information to development via email. The solution also automatically constructs high-level monitoring information for business transactions built from the detailed PurePath data it captures. This enables dynaTrace to provide business-relevant performance metrics to business or nontechnical consumers. (Some use dynaTrace in conjunction with application performance management and monitoring tools from HP Software, CA, and others.)

The heart of the dynaTrace system is its diagnostic server. As stated earlier, dynaTrace deploys agents to the server, and those agents instrument the target application. Once the end user makes a request to the application, dynaTrace’s agents record and immediately send performance and contextual information to the diagnostic server. Since this processing is done on the diagnostic server, there is typically nominal overhead, even when applications are monitored at very deep levels under production loads. Based on this approach, production teams are able to analyze data in real time (or offline) and triage issues as they arise. They also can drill directly into source code from the PurePath as needed and appropriate once root cause has been isolated.

The company has planned third-party integration — the technology currently can plug into IBM Tivoli and integrate with Eclipse and Visual Studio as well as the testing tools from HP and Borland. Some companies are proactively using dynaTrace for architectural validation, profiling, and performance in conjunction with Rational test for staging, performance tuning, and capacity planning. Proactive, up-front diagnostic information about existing problems improves code quality earlier in the application life cycle, such as architectural validation, performance regression testing, and white-box testing. This can assist companies with internally built as well as outsourced and offshored code analysis and validation.

**TOP CHALLENGES**

**Shifting Culture and Bridging Gaps**

Making the transition successfully to combined testing, monitoring, and diagnostics within the enterprise demands cultural change and process and organizational coordination. Although organizations are highly motivated to adopt effective diagnostic approaches when a business-critical application has unacceptable
performance, consistent benefits come with more consistent coordination and usage across teams earlier in the software life cycle. It is therefore key to combine adoption of coordinated tools with a shift in process and organizational approaches to gain earlier cost benefits and to optimize performance earlier.

With the advent of increasingly complex infrastructure services–based approaches to software creation and even less visibility "under the hood," it becomes all the more pressing to pull together effective processes with appropriate automated tools.

Linkage between the business side, development, QA, and IT for successful development, deployment in production, and a shift to more effective performance optimization provides obvious benefits over earlier, labor-intensive approaches to discovery of performance problems. Yet fractured organizational structures must be changed, since they typically result in disparate approaches and inadequate coordination between development, testing, and operations to isolate problems and repair them successfully. The business and cost benefits of a different approach and increased maturity provide incentive — but it's possible to provide metrics for return on investment (ROI) to justify expenditures and change only if the current baselines are captured. (All too frequently, organizations hide the costs and time frames required to address business-critical performance challenges.) One of the initial barriers to dynaTrace adoption for users was an initial perception on the part of management that pricing was too high. (dynaTrace has created new pricing models to address this issue now, however.) Creating baselines to measure cost benefits is something that must occur up front.

The requirement to also include effective transactional support in the face of significant complexity, monolithic systems, stovepipes, and other challenges helps to drive change and corporate agility. One of the key benefits of this type of technology is to provide visibility and to surface hidden issues in an accessible, intuitive manner. User case studies demonstrate how to make this transition and key areas of quantifiable business benefit for dynaTrace — discovery of application problems that could have remained unresolved and significant, rapid performance improvements. Key to these case studies is a transition to performance optimization from unmanaged or ad hoc approaches and the value to combining this approach with typical monitoring and testing tools for greater business adaptability.

CONCLUSION/RECOMMENDATIONS

Competitive organizations should combine development testing, production analysis, and diagnostics to enable high-performance transactional systems that are timely and of high quality. The costs of ad hoc guesswork to determine causes of poor application performance are untenable. The exponential cost increases that result from discovering problems late in the software life cycle are prohibitive in a difficult economic climate. Increasingly, organizations strive to cut costs and resource allocation. Up-front analysis and prevention of application problems in production is an excellent area on which to focus.
The benefits of managing application performance through detecting, diagnosing, and resolving performance bottlenecks enable companies to meet core customer service-level agreements (SLAs). Proactive planning that leverages diagnostics up front enables companies to avoid performance problems before they occur. This information can be leveraged within the development, testing, and production life cycles. Users also benefit from solving performance problems in the software itself rather than purchasing new hardware to try to optimize application access.

The costs of poorly performing or nonperforming business-critical applications are prohibitive. Users should evaluate and improve application visibility using diagnostics with an automated system across disparate teams, life-cycle tools, and deployment platforms.

CASE STUDIES

Centris Dramatically Improves Performance Time for Core Application

Centris, a European application service provider for health insurance companies, was changing to a new software solution along with new infrastructure. Performance for this backbone application was business critical since Centris supported the core transactional business for these health insurance companies — from medical invoices to claims processing and customer service — with that software.

Centris' new system was poised to go into production with the first customer during the fall of 2007. An outside supplier created this software release, which would also be running on new physical and virtual servers, a new operating system, and newer database technology. As the team began system deployment a few weeks before going into production, it wanted to be better prepared for analyzing and monitoring stability and performance of online processes and batch processing. The company had an application performance monitoring tool in place but found that its code analysis capabilities were insufficient to address these needs.

The company brought in dynaTrace despite significant financial advantages that were contained in an existing contract from a major supplier.

dynaTrace's product became a communication vehicle between Centris and the software supplier that built the software. Together, representatives from the software provider and people from Centris' IT services department used dynaTrace to analyze many problems and then solve them. Sample issues uncovered by dynaTrace included memory problems with the Java virtual machine (JVM), application exceptions in the software itself, problems with SQL statements running too long, and issues that they had not seen before during the development and monitoring process. Even though they hadn't discovered these issues as part of development and deployment, they ended up addressing the issues.
Dynamic Turnaround Time and the Customer’s Prior Experience

Centris had been doing software tests with load testing and other automated tools for months before going into deployment but had not identified all optimization potential. With dynaTrace, it was able to find and solve problems in four weeks that it hadn’t seen in the eight or nine months prior. dynaTrace saved tremendous time for the company and gave key insight and analytics that were unavailable during the earlier development cycle. In terms of improvements to the application’s batch performance, initially, 50–60% of transactions had response times of less than two seconds and almost half of the transactions took much longer than two seconds. Since Centris started using dynaTrace to diagnose the performance issues with the software and database suppliers, 96% of the transactions are now done within two seconds or less.

Centris also has the ability to be quickly responsive to emerging issues. When a customer tells the Centris service center about a performance problem, it is able to see immediately the response times related to the customer’s transactions. In most cases, the service provider is able to solve the problem within the same day and the customer is very happy with that — there’s very quick response time. For example, if the user tells the support team that it is using “X function” and that it takes five seconds, the support group has visibility into that problem and can see why it’s happening and, in most cases, solves it very quickly or can suggest improvements/modifications to the software supplier.

The most important point is that it was possible for the company to show the providers evidence in case of problems. Centris’ staff members also have visibility themselves to support their own direct clients when they have performance challenges.

Lessons Learned: Diagnose Sooner and More Often

Key takeaways from Centris’ experience would be to use a tool such as dynaTrace much earlier in the project and not merely a few weeks prior to production. Using the product at the beginning of the software development would have saved time, and the company feels it would have been better for the project if it had begun this process much earlier.

When Centris started the application analysis, the software was almost 99% completed. The company now has to live with structural challenges that it would probably have approached and designed differently if it had the insight earlier. Revamping the software at this point is cost prohibitive. Even though the company’s software supplier uses different analysis tools, dynaTrace provides an ongoing opportunity for clear communication between the service provider and its supplier to improve the product over time.

Deployment of the dynaTrace product was straightforward and completed within a day. The only thing that was a bit complicated initially was to find out which processes were the best points to measure and to then tune the configuration for dynaTrace. A dynaTrace specialist was able to do that within three days, and the dynaTrace product has been in production now for nine months. Centris continues to gather more and more information on the application and how its customers are using it to improve their dynaTrace configuration and benefit from the analysis. For this key initial customer in a growing market, support and training have been excellent.
Peopleclick: Saving Time and Cutting Costs with dynaTrace

Peopleclick, a human resources software as a service (SaaS) provider with 300 employees and over 1,000 clients, needed better application performance analytics. With hosted recruiting software applications on over 200 different sites and servicing over 1 million users, and with customer growth rapidly increasing its system traffic, Peopleclick simply could not afford the time it was taking to resolve performance challenges. During peak times, Peopleclick could have about 7,000 concurrent sessions with users interactively using the system and around 1.5 million page views per day. With an international user base accessing the software 24 x 7, this career site provider had little leeway for downtime or poor performance.

Peopleclick’s biggest pain points and primary ROI for spending the money and time on analysis came down to the mean time taken to resolve complex software issues. The company needed additional information to uncover details about key problem specifics. Attempts at discovery involved a combination of guesswork and code rework that could create additional problems without addressing the original issue(s). Without tools to clearly discover what was going on and where and why the exact problem was happening, resolving issues and getting to root cause analysis in a reasonable time frame were significant challenges for Peopleclick. The company already had TrueSight to monitor real user traffic coming into its Web applications. It could tell how many concurrent users were coming in and how that traffic trended over time. This allowed Peopleclick to see the HTTP level and understand user sessions, occurring errors, and error pages. Although this gave some visibility into what end users were experiencing, the company was missing the core ability to quickly find the root causes of certain complex software problems to resolve them.

One of Peopleclick’s differentiators is its ability to configure career sites that not only match a customer’s brand but also are tailored to its business processes. But in doing so, this can introduce additional complexity in the SaaS environment. Sophisticated solutions such as dynaTrace give Peopleclick the visibility to more efficiently narrow in on root cause in such a complex environment.

Key Functional Selection Criteria as Driving Factors

The selection criteria for an automated tool included multiple capabilities. Peopleclick was seeking a production profiler that could profile code under production loads continuously, 24 x 7. In addition, it needed a product that would enable high degrees of specificity about what was being profiled, such as specific method calls and classes. The company required a product that would be intuitive to use for both developers and operations teams. Categorization and naming capabilities were important in that context. The company wanted a solution that would make it easy for both development and operations teams to obtain, understand, and share the data. Plus, it wanted to be able to share the data results with partners, teams, and clients,
as appropriate. In addition, as a multiplatform shop, Peopleclick required support for both Microsoft .NET and Java with the same product, the same interface, and the same install for both environments.

Peopleclick found that dynaTrace addressed all of these areas and also helped proactively identify potential and pending software problems. Because dynaTrace continuously monitors code behavior with a rolling buffer, it can gather every page request and all instrumentation points. If something "interesting" or divergent occurs (e.g., if a page takes more than 10 seconds to load), dynaTrace files it and saves it for viewing. This enables users to identify potential problem areas before they actually become excessively problematic and obvious to clients. (This alerts developers and teams early in the process.) Users can also clearly specify an area for profiling (for instance, "If this method for this class gets called, I want to be notified" or "If this specific method raised this error or exception, I want to know about it"). In this way, users are able to instrument dynaTrace with high degrees of specificity.

dynaTrace's intuitive user interface is particularly helpful to Peopleclick staff, enabling both developer and operations teams to make sense of dynaTrace data and get to the bottom of software issues. This is helpful even for laypeople who don't understand the internals of applications and who are then able to figure out a problem without having to escalate the issue to a technologist. Peopleclick found that these tools began to blur the line between tools that target operations, development, or users. dynaTrace provides the ability to label APIs with meaningful names to categorize information to help operations teams.

Since Peopleclick has a SaaS model for software delivery of its recruitment application, data privacy is key with stringent policies in place. dynaTrace allows the company to package up incident data and share it via dynaTrace Explorer. In this way, users are able to share information with partners, clients, and others to let viewers load profiler data to slice and dice the information. With other products evaluated by Peopleclick, sharing information required a clunky HTML export, resulting in multiple pages of challenging information. And with dynaTrace, users don't have to be tethered to a console.

**Transition to dynaTrace: Core Time and Efficiency Savings**

Peopleclick found the dynaTrace implementation to be straightforward. The biggest effort involved was the actual tuning of the product to configure it for Peopleclick's particular situation and determining what needed to be profiled and measured. Peopleclick's team found that the sales engineers at dynaTrace facilitated that process in helping to target the solution appropriately. Peopleclick found the dynaTrace engineering support to be excellent in that context.

Although Peopleclick is still in the early phases of using dynaTrace, it already has obtained significant benefits.

The company has already proven that it can find and take action on potential problems before customers even notice them. If an issue occurs in one out of 100 or even 1,000 transactions and it isn't investigated, it is likely to eventually become problematic. Peopleclick is now finding issues with enough lead time to take action before it has a significant problem.
One of the most important benefits to Peopleclick is in the time savings to both discovering problems and resolving them. Certain issues Peopleclick faced required insight into the internal workings of the JVM. Before bringing in dynaTrace, Peopleclick could only speculate as to what was going on in the JVM. It would then make changes around what it assumed might be the culprit. Sometimes the assumptions were correct and sometimes not, thereby wasting development, QA, and deployment cycles.

This time cost is not just calendar time to resolution, but significant multipliers on staff time to guess about code changes — you don’t want to make those “fixes” on a whim. With dynaTrace, Peopleclick can help target its time more wisely. It is able not only to explicitly target and uncover the root problems for software issues but also to discover challenges before they become external problems.

From a financial perspective, the costs of additional and potentially unwarranted software releases can be significant.

Another benefit of dynaTrace is its ability to enable collaboration across teams. dynaTrace’s PurePath is very cut and dried with the facts — it identifies the problem and shows the context. (Developers can take the file, see the issue, and slice and dice the data themselves.) By sharing PurePaths, development and operations teams can better collaborate and determine the root cause of problems.

**Key Takeaways**

Peopleclick is pleased with the process that it went through in evaluating multiple vendors (four to five), choosing and adopting dynaTrace, and realizing cost and efficiency savings and improved customer support. The company would encourage users to establish key functionality criteria and to make the vendors work alongside the evaluators during that process to be certain the product does what the vendors say it will. Peopleclick’s dynaTrace users have found that it is intuitive and very well-designed, works well, has strong features throughout, and is well-documented. But most importantly, the company is now able to be proactively responsive with its software solution. Given the demands of the HR recruiting vertical market with multiple product lines and hiring managers, the ability to effectively meet SLAs and optimize performance is business critical.