

“dynaTrace proves to be just what the doctor ordered”

By Marc Pinder, Senior System Designer, November 07

“In 2008 we will face a number of challenges - one is to ensure the smooth and trouble-free implementation of a major upgrade to our UK insurance systems. It’s important that we are able to get ‘under the skin’ of our new system in order to see what is really going on. Our aim is to undertake detailed code reviews as early in the software development process as we can. This way we don’t have to wait until the testing phases to unearth issues with application design which could lead to poor performance and scalability when we go-live. The earlier we capture fundamental design issues, the cheaper they are to fix.”

When looking at products to help us improve performance we started with ‘profiling tools’. Unfortunately we found that they only go part of the way in assisting us with our application code design reviews - crucially they aggregate data and whilst they can identify obvious bottlenecks they run out of steam when attempting to diagnose more complex issues. One limitation we found is that they can’t help us understand why a particular function can, on one day, be called 500 times and perform flawlessly in less than a second, but occasionally takes over 30 seconds to execute. Profiling tools simply don’t gather the historical detail or context of exactly what was going on when the problem occurs. This is why we were immediately interested in dynaTrace Diagnostics when introduced to it by Application Performance.

dynaTrace Diagnostics has the key concept of a ‘PurePath’ which traces a particular path of execution from a defined point in the system. It presents this PurePath along with all the classic profiling information needed in order to build a live call stack view for individual use cases of business transactions. dynaTrace Diagnostics can also span physical tiers unlike any other product we have come across, allowing us to trace a transaction across boundaries. This allows us to understand all the components used and how much time is spent in each all the way from our Java web client through the .NET application tier to the backend Oracle database cluster.

With this unique functionality we have been able to use dynaTrace Diagnostics to review .NET and Java systems even where we have no source code. We are now aware of how the application code works in detail including understanding how the different tiers connect and interact even if they are on physically separate machines. We can also trace true end-to-end processes which involve more than one system.

Another important feature of dynaTrace Diagnostics that we use a lot is its ability

to drill down through related views of the method, web-service and database calls all of which can be

output to an incredibly useful UML sequence diagram. We have now started to save these diagrams and use them as a quick way to document our application design.

Prior to buying dynaTrace Diagnostics we conducted a two day proof-of-concept. The install was remarkably easy taking less than 10 minutes and from this impressive start we were able to identify a number of performance issues in a matter of hours. These included: database calls from the wrong layer, redundant code, and excessive method calls. By the time the proof-of-concept was complete it was clear that we had a gap in our existing toolsets and needed to fill it with dynaTrace.

We are now using dynaTrace Diagnostics to directly influence our application code review process. We typically start with ‘business processes’ and use the tool to understand exactly how it executes. We then take the output from dynaTrace Diagnostics, including the live call stack and parameter data values, and send them directly to the development team to assist with the fix. Needless to say this is helping us save time and also work more closely as a team.

Looking beyond design reviews, dynaTrace Diagnostics is starting to interest us for the production environment - with minimal overhead, dynaTrace agents can be switched on in production to provide real-time diagnosis of issues which only occur in production. Triggers can be used to record PurePaths in the event of particular issues: a method taking a long time to execute, a user executing a key business process, etc. Our support teams can then pass them back to development for resolution and because everyone will be using the same tool this can be done very efficiently.

In summary, we believe that through the use of dynaTrace Diagnostics our production and application support teams will be much better placed to spot and resolve problems earlier in the lifecycle than otherwise possible.