What is the problem/issue you are addressing?

Today’s internet enabled enterprise applications tend to be quite large and complex and often developers find it difficult to maintain a good understanding of the overall system behaviour. This comes from the fact that current systems no longer conform to a large monolithic architecture and instead have been moving toward highly interconnected, multi-tier distributed architectures designed to run on a heterogenous collection of servers. During system testing it can prove extremely difficult to piece together logs from different servers to form a coherent picture of the entire application. Understanding system behaviour in such circumstances can be difficult, as developers/testers are required to sift through and correlate a range of different log files in an attempt to understand how the different components that make up the system interact.

Furthermore, detailed documentation on such large complex systems is often unavailable. While documentation is often part of the early design phase, it is common that, as the system evolves during the many iterations of the development cycle, the documentation is not updated. Thus the documentation, which had been drafted early in the development cycle, may no longer reflect the current system. Again this can be a major problem for developers analysing a system (e.g. debugging problems, analysing the design for flaws etc.) since no up to date documentation is available to help them reason about the system behaviour.

Why is this important for IBM or IBM’s customers?

System comprehension and up to date documentation is essential to allow for high level discussions/analysis of large systems. Often such analysis can reveal major design flaws that can seriously affect the system in terms of its performance, maintainability, reusability etc.

In this talk we will present an approach for extracting end to end UML class and sequence diagrams from a running system. The approach makes use of the COMPAS JEEM open source monitoring tool. COMPAS JEEM can monitor any J2EE application and collects the different call paths through the system. The output from the tool is in the form of end to end run-time call paths that show how the different components that make
up the system interact to service the different user requests. A run-time call path shows how a user request is serviced, giving the ordered sequence of events/calls and related performance and resource usage information. They can be used for a number for different purposes in the area of system comprehension (e.g. problem determination, design analysis, performance analysis, reverse engineering etc.).

**What are the key results?**

Using our approach we have reverse engineered and analysed a number of J2EE applications. Our results focus on 3 applications in particular. Two sample applications and a real IBM application (Workplace).

Our reverse engineering phase involved collecting the different call paths that make up the application using COMPAS JEEM and producing UML diagrams (class and sequence) from the output. While the UML diagrams were produced manually, this process can easily be automated.

Next we analysed the UML diagrams to identify performance design flaws (i.e. performance antipatterns) that might exist in the applications. We will present our approach in detail at this talk including the different antipatterns (and potential antipatterns) found in the applications. We will also go into detail on how the COMPAS JEEM monitoring tool captures the call path information.

**What is new in this work (compared to the existing IBM internal/external work)?**

COMPAS JEEM is an end to end call path tracing tool for J2EE systems. It is non-intrusive since neither the source code of the application nor the source code of the application server need to be modified. It performs the tracing by using standard J2EE mechanisms and thus is completely portable. It can be applied to any standards compliant J2EE web server/application server/database combination and can trace calls from the web to database tier, thus capturing the end to end behaviour of the system. It also has the ability to trace calls across distributed JVMs. It has been tested on a number of different application server/database combinations including (e.g. Websphere, Jboss, Oc4J, MySql). IBM currently provides no tools to collect end-to-end call paths for J2EE systems.

**What do you propose to do with these results?**

As far as we are aware Websphere has no built in mechanism for intercepting calls to J2EE components (except for JSPs/Servlets). This capability is provide by COMPAS JEEM. IBM also does not provide any tools for tracing system wide J2EE call paths. COMPAS JEEM can be used with Websphere to provide such features. We plan to investigate how the tool might be integrated into Websphere to provide these features as standard. We believe the Websphere Work Area Service could potentially be used as part of such a solution.
Future work includes automating the diagram creation process and automating the antipattern detection process. We will also be presenting this approach and results to the Performance and System Test teams in Dublin.