A major issue for today’s enterprise applications is that they frequently fail to meet their performance requirements on time or within budget. That is these systems can be very large and complex, and often developers do not have a complete understanding of the overall system design and system behaviour. Current performance tools do not help greatly, since they (generally) only monitor the running system and present vast amounts of information to the tool user. We believe there is a real need for more advanced performance tools. Such tools should perform analysis of the data collected, and in turn make sense of this data, as well as present it to the user in a more meaningful format. In this work we present an approach for proactive problem determination of poor performance design in enterprise systems. Our approach can automatically detect performance design and deployment antipatterns through run-time design extraction and advanced analysis techniques.

In this presentation we will discuss how the design of JEE systems can be automatically extracted from data collected at run-time. The extracted design captures a number of important relationships that can be further analysed for automatic performance antipattern detection. The approach works by monitoring the system to capture the following information: (a) run-time paths, which contain the ordered sequence of events that are invoked for a particular user action as well as related performance information, (b) object usage information, which captures how particular objects are created and accessed along the different run-time paths (c) server resource usage information (e.g. container thread pools, database connections etc.) and (d) component meta-data. Our work shows how this information can be obtained in a portable manner. The data captured is analysed offline to extract interesting relationships and patterns that are used to construct a design model of the JEE application. During analysis the following information is extracted from the monitored data and added to the design model: (a) component relationships, (b) component communication patterns, (c) object usage patterns, (d) reconstructed run-time container services, (e) path clusters and (f) statistical performance metrics. We discuss a number of advanced analysis techniques that can be used to extract this information, including a number of techniques from the field of data mining. A rule engine can be loaded with antipattern descriptions and can be applied to the extracted model to identify well known JEE performance antipatterns. We show how rules can be easily written to describe antipatterns in a concise manner. We have applied this approach to a number of
JEE applications (including the IBM Workplace application) and have automatically identified a number of JEE performance antipatterns.

During this presentation we will also discuss the different antipattern types and in particular we will show how JEE performance antipatterns can be categorised into a number of different groups. The antipatterns are grouped based on the information required to detect them.