The Problem: What and Where are the Performance Issues?

Changes in the number of simultaneous users or changes in user-initiated transaction distribution can yield the system performance unacceptable. The use of complex component-oriented middleware and COTS components makes it difficult to understand and predict performance issues.

COMPAS: Identifying Transactions

Having determined the transaction tree, the modeling module can improve the monitoring process. Only the transactions that have components with alerts will be monitored to reduce overhead.

COMPAS: Performance Modelling and Prediction using the Model Driven Architecture (MDA) Approach

Having a high-level view is not enough to understand the precise nature of the problem. For it in a system.

Our Solution: The COMPAS Framework - Overview

The Component Performance Assurance Solutions (COMPAS) framework addresses the performance and efficiency aspects of component-based systems. It is structured into three interrelated modules:

- Monitoring: gathers real-time performance information from a running application, non-intrusively.
- Modelling: creates UML models of the target application using information from the monitoring module, augmented with UML performance indicators.
- Performance Prediction: the generated models of the applications are simulated with different workloads.

COMPAS: Monitoring EJB applications

The monitoring module uses monitoring components to automatically generate a parallel application, which mirrors the target application that must be monitored. For each component in the original application (target component), a Proxy component is inserted in the same remote interface and uses the same Java Naming and Directory Interface (JNDI) name, exists in the mirror application.

A proof-of-concept monitoring module has been implemented for the EJB platform, using Java Management Extensions (JMX). It can display hierarchical graphical consoles showing real-time data.

Each scenario is represented by a test case. The first scenario starts with step 1.2, “addItem” and the second scenario starts with step 2.2, “addReview”. In the example diagram, scenario 2 has a probability of occurrence of 0.2 and an average execution time of 100ms. One of the steps in scenario 2, step "updateMarketingDB" has an associated mean execution time of 200ms. One of the steps in scenario 2, step "updateMarketingDB" has an associated mean execution time of 200ms. One of the steps in scenario 2, step "updateMarketingDB" has an associated mean execution time of 200ms.

A high-level view of the model driven architecture framework can be "zoomed in" to that precisely determine the origin of the performance issue.