

Add ARM performance monitoring easily with Eclipse

Extend enterprise management tools directly to your applications

Skill Level: Intermediate

[Ashish Patel \(ashishp@ca.ibm.com\)](mailto:ashishp@ca.ibm.com)

IBM Performance Optimization Toolkit (IPOT) Architect and Lead Developer
IBM Rational

[Oliver E. Cole \(oec@ocsystems.com\)](mailto:oec@ocsystems.com)

President
Freelance Consultant

06 Feb 2007

The Eclipse Test & Performance Tools Platform (TPTP) project released an open source and Application Response Measurement (ARM) V4.0-compliant implementation in June 2006, based on TPTP V4.2. This tutorial describes the TPTP project and explains how ARM has been implemented and integrated with TPTP. The trade-offs involved in getting to this point are discussed, along with the challenges in moving forward. Specific examples show how to get started using ARM for your application with TPTP.

Section 1. Before you start

This tutorial describes how the Eclipse Test & Performance Tools Platform (TPTP) project is extending its toolset to use The Open Group Application Response Measurement (ARM) instrumentation methodology for response-time tracking. Starting with TPTP V4.2, performance analysts will be able to use Eclipse TPTP -- or products based on TPTP technology -- to identify and monitor individual transactions as they pass through the various components of composite applications.

The implementation of ARM described in this paper was released as a Technology Preview with TPTP V4.2 in June 2006. In the future, it will be fully integrated into TPTP.

Section 2. About Eclipse and TPTP

Eclipse is an open source community whose projects are focused on providing a vendor-neutral open development platform and application frameworks for building software. The Eclipse ecosystem is built on royalty-free technology that offers a universal platform for tools integration.

All technology and source code provided to and developed by Eclipse is made available royalty-free via the Eclipse Public License. As a result, tools vendors can and are integrating Eclipse technology into their products.

The Eclipse Platform is written in the Java™ programming language and comes with extensive plug-in construction toolkits and examples. It has already been deployed on a range of development workstations including Linux®, HP-UX, AIX, Solaris, Mac OS X and Windows® systems.

The Eclipse Test & Performance Tools Platform² (TPTP) is one of many Eclipse projects. TPTP provides an open platform that allows software developers to build unique test and performance tools, both open source and commercial, that can be easily integrated with the platform and with other tools.

TPTP addresses the entire test and performance life cycle, from early testing to production application monitoring, including test editing and execution, monitoring, tracing and profiling, and log analysis capabilities. The platform supports a broad spectrum of computing systems, including enterprise, high-performance, stand-alone, and embedded systems.

The Tracing and Profiling Tools Project is a subproject of TPTP that provides a framework for building tools that collect and analyze application performance information. The project includes tools that collect trace and profile data for single-system Java applications through the JVMPI monitoring agent. It also includes Build-to-Manage (BtM), a generic toolkit for instrumentation using ARM, Java Management Extensions (JMX), and Common Base Events (CBE).

V4.2 of Eclipse TPTP implements a version of the Application Response Measurement³ specification, which can be used to instrument specific points in an application and collect trace and profile data for a distributed Java application while delivering more accurate profiling measurements.

Section 3. About ARM

The ARM standard was developed by The Open Group, a vendor- and technology-neutral consortium that strives to enable access to integrated information within and between enterprises based on open standards and global interoperability.

The ARM standard describes a common method for integrating enterprise applications as manageable entities. The ARM standard allows users to extend enterprise management tools directly to applications, creating a comprehensive end-to-end management capability that includes measuring application availability, application performance, application usage, and end-to-end transaction response time.

Applications make calls to application programming interfaces (APIs) defined by The Open Group when transactions begin and end, allowing these transactions to be measured and monitored. This information can be used to support service-level agreements and to analyze response time across distributed systems.

ARM instrumentation has grown in popularity since it was first introduced in 1996 and is now built into software from leading vendors such as IBM®, Hewlett Packard, SAS, and Siebel. Software that is not already ARM-enabled can be made to have calls to the ARM API embedded directly in the source code, or calls can be inserted into the machine code at runtime.

Section 4. Getting started

This section describes step by step where to get the current version of the TPTP ARM capability, how to download and install it, and how to get ARM data from a demonstration application.

To download and install Eclipse TPTP and the Tech Preview with the ARM instrumentation:

1. Point your browser to [Eclipse Test & Performance Tools Platform \(TPTP\) Project](#).
2. Click **Downloads** .

3. Select **4.2** from the tabs at the top of the page.
4. Select a **Stable Build** of the version.
5. Download the following components:
 - a. TPTP Runtime (SDK is optional)
 - b. Agent Controller
 - c. Technology Preview Section: ARM Instrumentation (ARM Engine and ARM UI)
 - d. All required components (including WTP, GEF, and JEM)
7. Install the workbench:
 - a. Unzip the TPTP Runtime, ARM UI, and all required components into the same folder.
9. Install the agent controller (a folder separate from the workbench):
 - a. Unzip the Agent Controller
 - b. Run <RAC_INSTALL_DIR>\bin\SetConfig.bat (or equivalent)
 - c. Unzip the ARM Engine into the same folder as the agent controller

[Installation guides](#) explain how to download and install TPTP, is available at Eclipse.org.

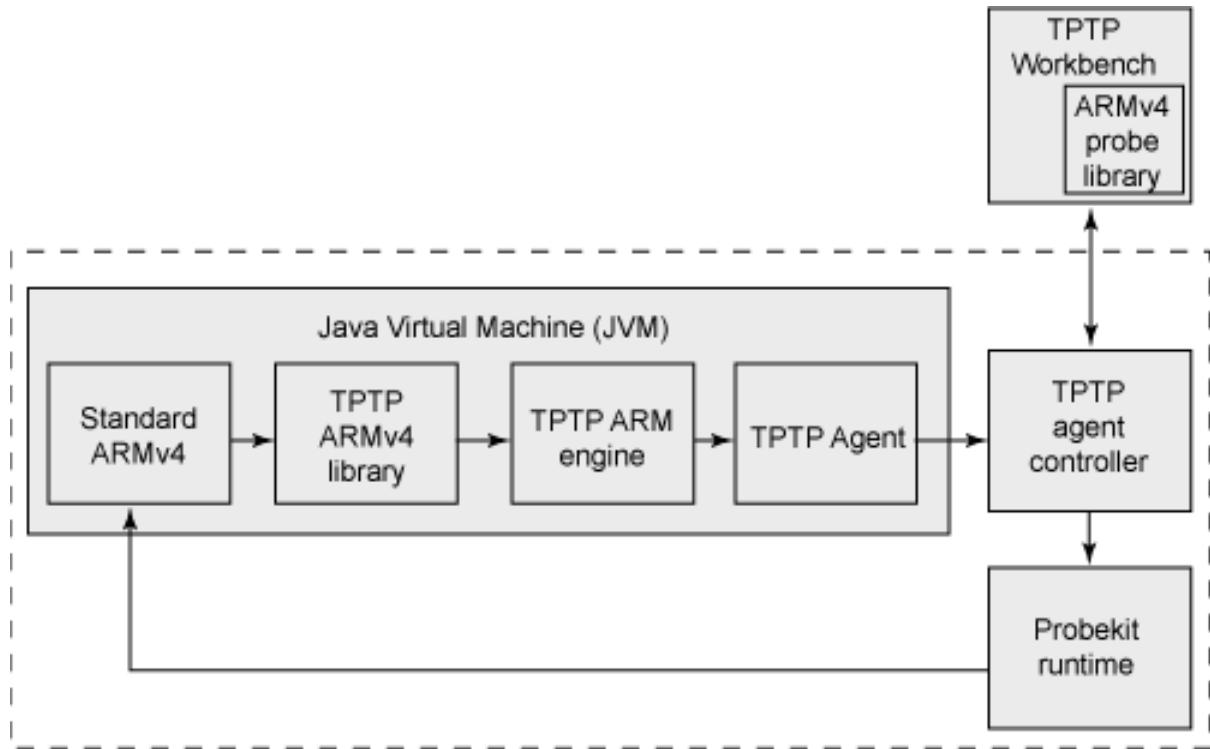
Once TPTP and the Tech Preview are installed, the user is ready to begin using ARM. The software to run this demonstration application is available for download from OC Systems (see [Resources](#)). A Flash version of the demo is also available there.

Section 5. Architecture in Eclipse TPTP

Using the ARM implementation in TPTP, a user can take any Java application and use TPTP to instrument ARM calls into application source code, collect the ARM data while the application is executing, and view the collected data with graphical viewers.

Figure 1 shows a block diagram of the TPTP ARM Data Collection Infrastructure (TADCI), which consists of the components shown.

Figure 1. TPTP ARM Data Collection Infrastructure (TADCI) Block Diagram



TPTP ARM Data Collection Infrastructure (TADCI)

TPTP ARM library

The TPTP ARM V4 library is an implementation of The Open Group ARM V4 standard. Instrumented applications make calls to the Standard ARM V4 interface, which invokes the TPTP ARM V4 Library and reports events to the TPTP ARM Engine.

TPTP ARM Engine

The TPTP ARM Engine processes calls made to the Standard ARM V4 interface and produces event-level information that outputs an ARM event for each event during a transaction. The TPTP ARM Engine communicates to a TPTP Agent, which utilizes the Agent Controller (AC) for communication. The output from the TPTP Agent is in an XML format, which is a human-readable, open document format that can be processed by TPTP Event Loaders.

To impact the performance of the application as little as possible, the ARM Engine is designed to operate on a separate thread in the same process as the application's Java Virtual Machine (JVM). Running the ARM Engine in a separate JVM process

could have skewed or misled an analyst when viewing performance results since the JVM itself is executing the application under investigation. In addition, this design approach allows the application to manage ARM Engine's life cycle.

TPTP workbench

The TPTP Workbench contains UI tools to:

1. Launch the application being profiled
2. Byte-code instrument applications to be profiled
3. Deploy Probekit source files to systems where the application is executed

The UI allows the application to be instrumented with a user-defined filter. The filter is applied at the instrumentation level, such that only those points in the application that match the filter's pattern are byte-code instrumented.

TPTP Agent Controller

The TPTP Agent and TPTP AC provide the communication mechanism between the ARM Engine and the TPTP Workbench. Data is reported from the ARM Engine to the TPTP Agent, which transfers it to the AC. The AC forwards the data to the TPTP Workbench. Though this data flow sounds complicated, the TPTP Agent and AC are transparent to users of the TPTP ARM Engine.

Probekit runtime

Instrumentation of the application is accomplished with the Probekit runtime, which accepts compiled Probekit files and uses byte-code instrumentation (BCI) techniques to instrument the application. The runtime is supplied with a prepackaged Probe Library, which contains a generic ARM probe that will instrument the entry and exit site of a method. All methods that pass a user-defined filter will be instrumented using this probe.

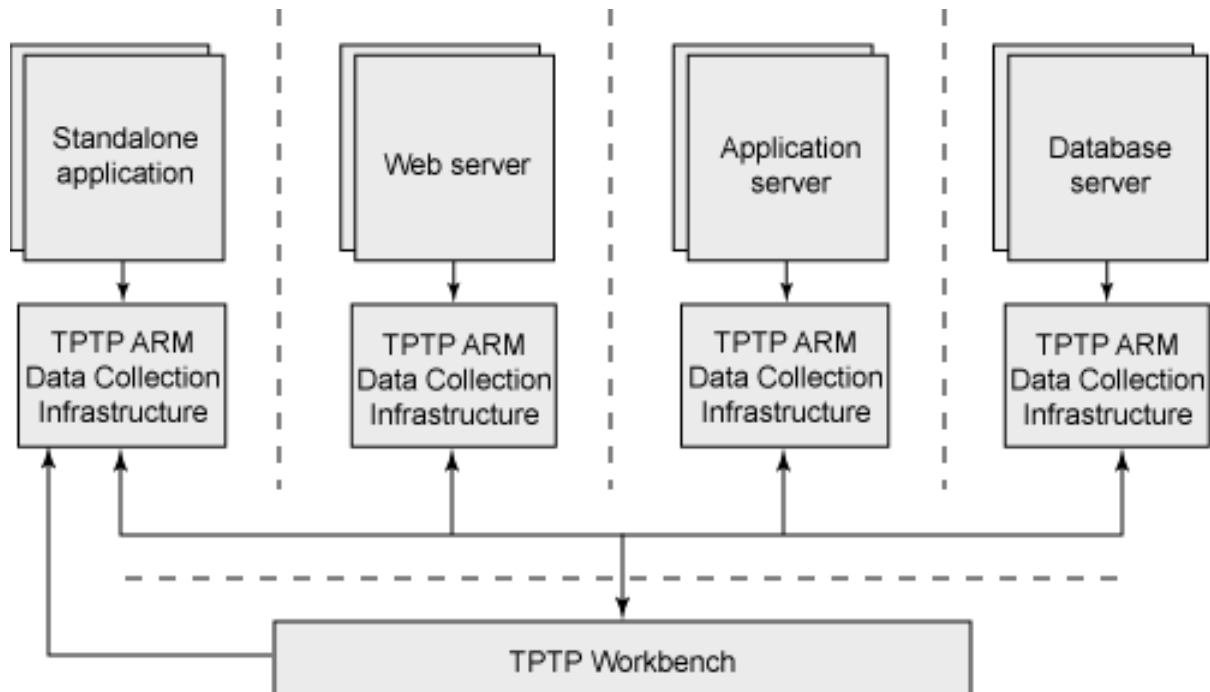
The following items are bundled with the existing AC, which includes the Probekit runtime:

- J2EE Probekit Library (also known as the ARM V4 Instrumentation prepackaged Probe Library)
- TPTP ARM Engine
- TPTP ARM V4 library

Distributed environments

When monitoring a single or distributed application, the user must install and configure the TPTP AC and TADCI on every machine where the application executes. The TPTP ARM Engine provides the functionality for dynamically discovering systems involved in a business transaction. Once a system is discovered, the ARM Engine will automatically have the TPTP Workbench attach to the discovered system. Figure 2 illustrates the TADCI in a distributed environment.

Figure 2. TPTP ARM Data Collection Infrastructure (TADCI) for distributed environment



To correlate transactions that occur on different processes and machines in a distributed environment, the ARM standard calls for the use of an ARM correlator. A correlator is generated for every root or edge transaction and each of its child transactions. A business transaction is determined by building a tree using these correlators. This process allows the ARM Engine to trace the path of a distributed transaction through the infrastructure.

To transport the ARM correlator across application servers in a distributed environment, the instrumentation flows the correlator as part of a CORBA Portable Interceptor when a distributed call is executed. Portable Interceptors are objects that an ORB invokes in the path of an operation invocation to monitor or modify the behavior of the invocation transparently. For our purposes, a Portable Request Interceptor is created and used to marshal and de-marshal the correlator during distributed transactions. The user is required to install the request interceptor in his application server.

Section 6. Steps to ARM-instrument an application

There are three basic steps involved in using ARM on an application with Eclipse TPTP:

1. Use BCI to insert the ARM calls into the application
2. Execute the instrumented application
3. View and analyze the results

Instrumentation

The first step to instrument an application requires inserting calls to the ARM API at strategic points in the application code. The Build-to-Manage (BtM) Toolkit for Java Instrumentation provides a wizard you can use to add instrumentation to selected packages, classes, and methods. The toolkit, originally developed by IBM Tivoli®, has been released to open source with Eclipse TPTP V4.2. The toolkit offers two ways to add the instrumentation: Probekit and AspectJ.

Probekit

Probekit is a scriptable BCI framework used to write Java code fragments inserted into an application to provide information about the program as it runs. This method instruments the classes of the actual application, rather than the application's container.

Probekit works well with the Eclipse TPTP framework when the application under test is started in controlled mode. Controlled mode halts the execution of the application until a client attaches to the profiling agent. Starting an application server in this mode instantiates the applications in controlled mode automatically, eliminating the need to restart the application server after its applications have been instrumented.

This mode allows the AC to monitor the application server without the TPTP Agent commencing processing until the Eclipse Workbench is instructed to monitor some aspect of the application and, therefore, having the TPTP Agent loaded will not affect the application server's performance significantly.

Once instrumentation probes are deployed to the AC, the probes will be applied to the classes that match the user-defined filter specification as they load in the JVM.

A user can create multiple probes and deploy any combination of these defined probes to the Probekit Runtime. This instrumentation mechanism uses the Class Load hook in JVMTI for dynamic instrumentation of applications.

Today, there is no mechanism to "undeploy" a probe. Therefore, once a set of probes is applied, the user must restart the application server because data from these probes is applied to the JVMPI agent and the Probekit Runtime. Users are required to utilize the **start** and **stop** buttons on the UI to control these. However, research is ongoing, and future releases of TPTP may allow probed classes to be undeployed. This behavior could be as simple as notifying the Probekit Runtime not to instrument new instances of the instrumented classes in the application when they load, and, hence, one can force the JVM to redefine the class -- in effect, undeploying the probes.

Static instrumentation

Probekit (as of TPTP V4.1) supports static instrumentation. During static instrumentation, Probekit rewrites the Java classes that are instrumented by adding invocation to the probe classes, which are added to the project. The instrumented application can run on any JVM, and there is no need for a Probekit runtime. All that's required is to have the probe classes available and in the classpath of the program.

Dynamic instrumentation

Probekit (as of TPTP V4.2) supports dynamic instrumentation. Dynamic instrumentation works similar to static instrumentation, but it performs the instrumentation in memory at class load time, instead of rewriting the class file on disk. The inserted byte codes are the same as in the static instrumentation case.

There are some consequences of using dynamic instrumentation. The called methods are loaded by the boot class loader and, therefore, in the instrumentation fragments one cannot refer to classes loaded by higher-level class loaders.

AspectJ

AspectJ is an aspect-oriented programming (AOP) implementation for Java technology in Eclipse. AOP allows programmers to create special-purpose modules, called aspects. An aspect simplifies the problem of adding consistent behavior to various locations in application modules. The set of points at which an aspect is applied is called a *point cut*. Advice is the code invoked at these points. Without AOP, code implementing a specific purpose, such as instrumentation of code, is scattered throughout application modules, making it difficult to add, remove, or modify.

The user then defines a point cut of method entries and exits. When the aspect is applied to the application at start, the resulting ARM calls profile the application.

Probekit and AspectJ instrumentation mechanisms work similarly. The only difference is that during instrumentation, Probekit uses a probe file while AspectJ uses an aspect file. The aspect file (aop.xml) defines several point cuts, or locations in the application code where instrumentation is to be injected. An example of a point cut is shown in Listing 1.

Listing 1. Sample point cut

```
<pointcut
  expression="
  execution(* com.ibm.ui.Client.*(..)) ||
  execution(* com.ibm.core.Library.*(..))"
  name="monitoredOperation" />
```

Using the Build-to-Manage Toolkit

To use the Build-to-Manage Toolkit (BtM), first install Eclipse TPTP. Support for ARM instrumentation is in TPTP Trace and Profiling Tools Project. BtM also supports Java Management Extensions (JMX) and Common Base Events (CBE), both of which are in TPTP Monitoring Tools Project.

BtM uses a preference page to determine whether to instrument with AspectJ or Probekit. At install, if the AspectJ Development Tools plug-in is present, the preference is set for AspectJ. Otherwise, the preference is set for Probekit. The BtM wizard is invoked through the Run-Profile menu. The toolkit offers the choice of instrumenting with ARM, JMX, or CBE. To use ARM, use the ARM Analysis Type when profiling the application.

Section 7. Run the instrumented application

To run the ARM-instrumented application, simply use the Profile button in Eclipse. Then create a profile launch configuration, the same way as done in TPTP today. The following sample shows how to profile a HelloWorld application.

Figure 3. Sample HelloWorld application

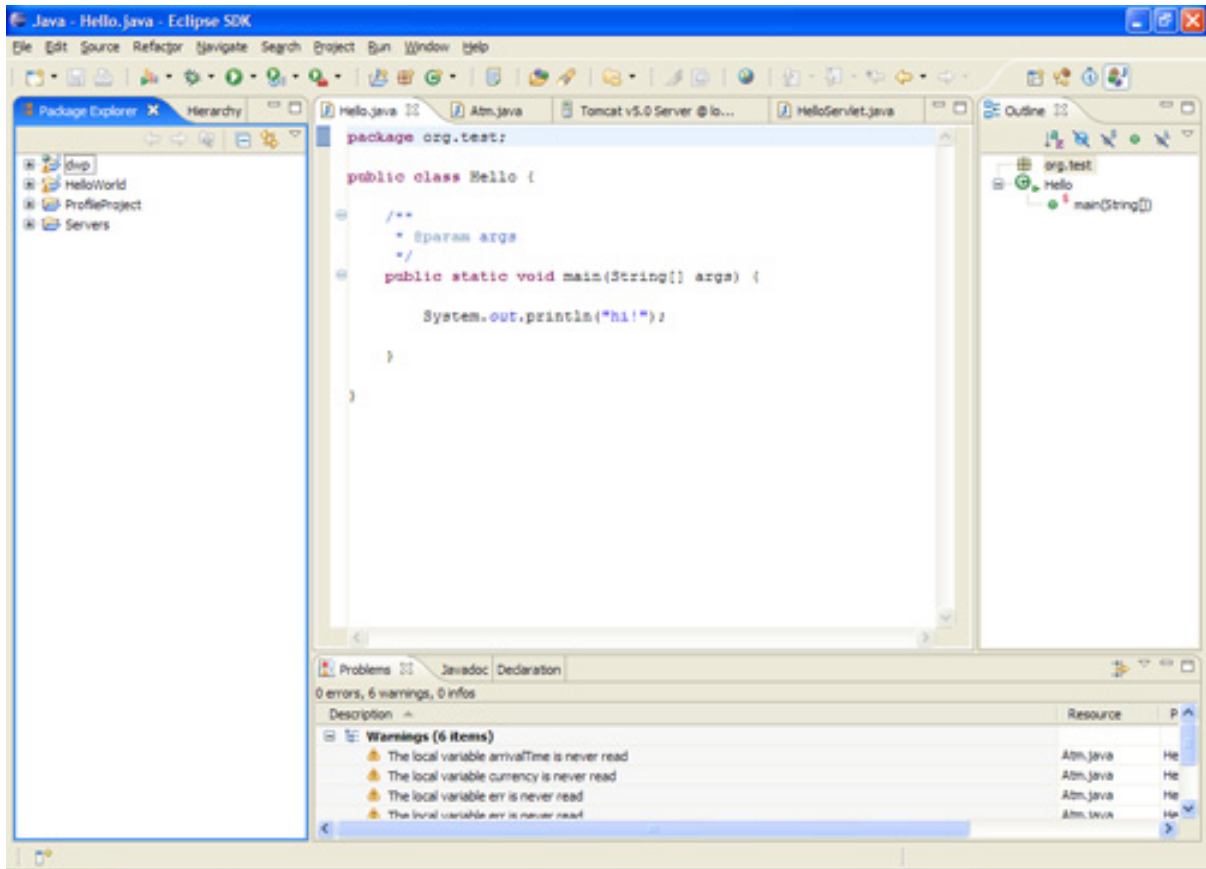


Figure 4. Open Profile Launch configuration and create configuration for sample Java application

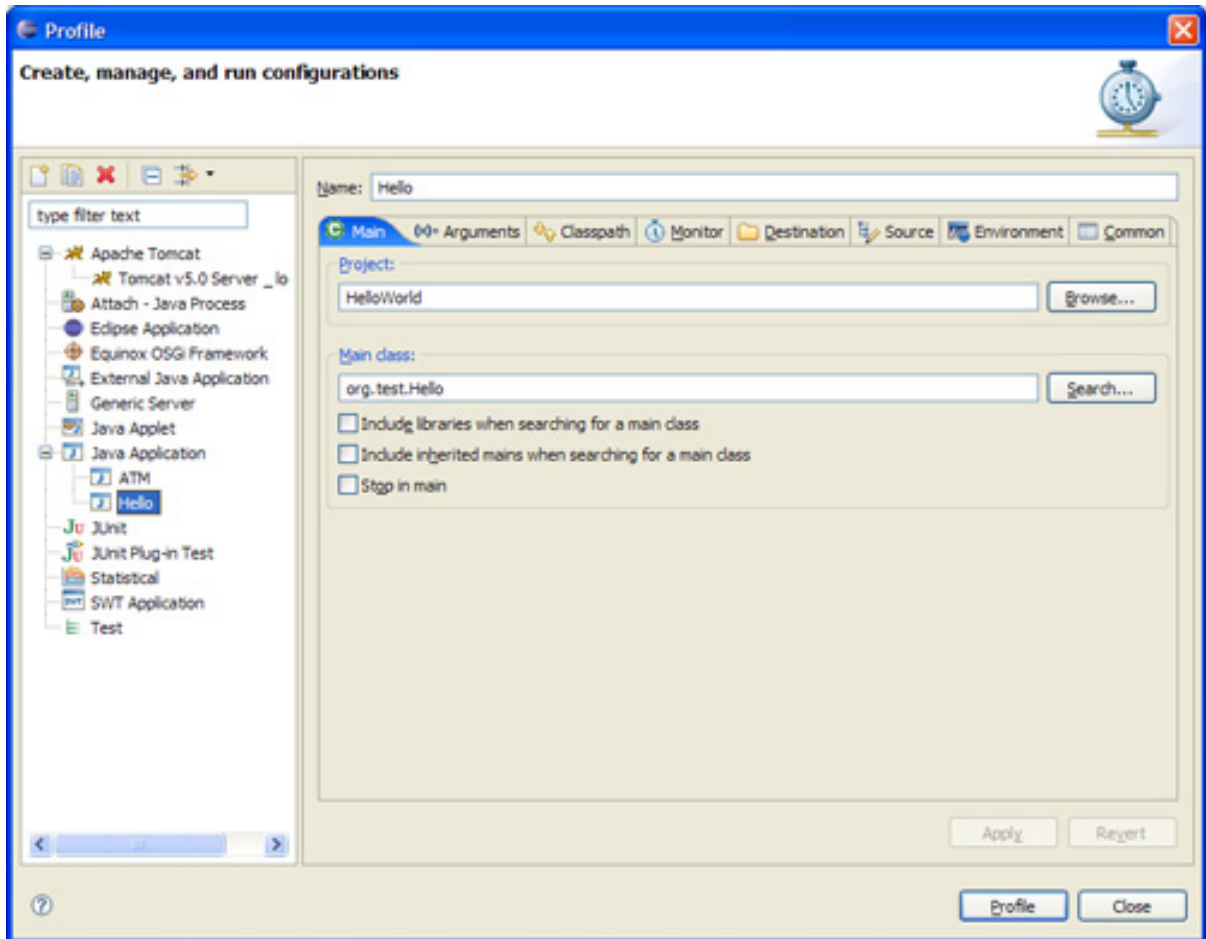


Figure 5. Select Instrument Collector as data collector for profiling application

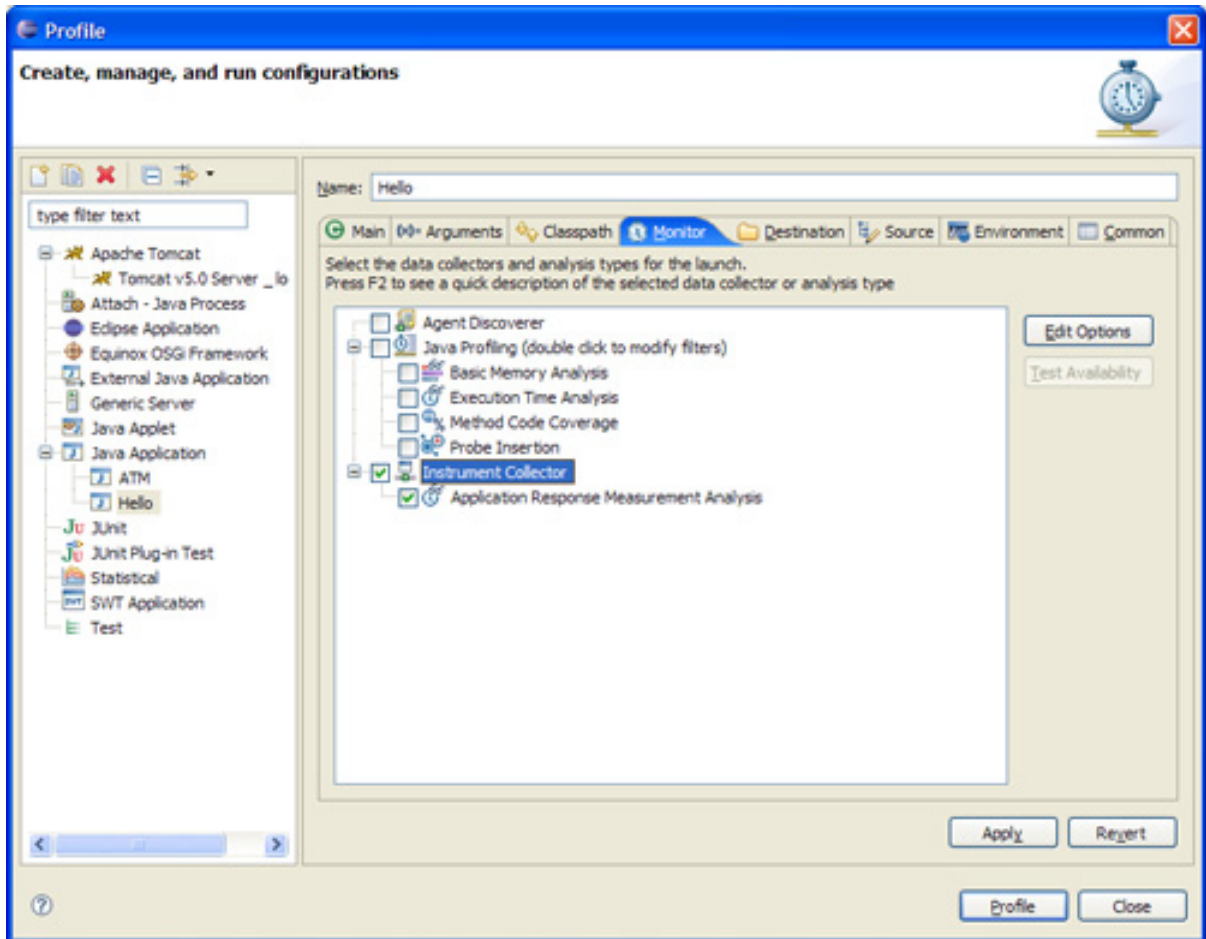
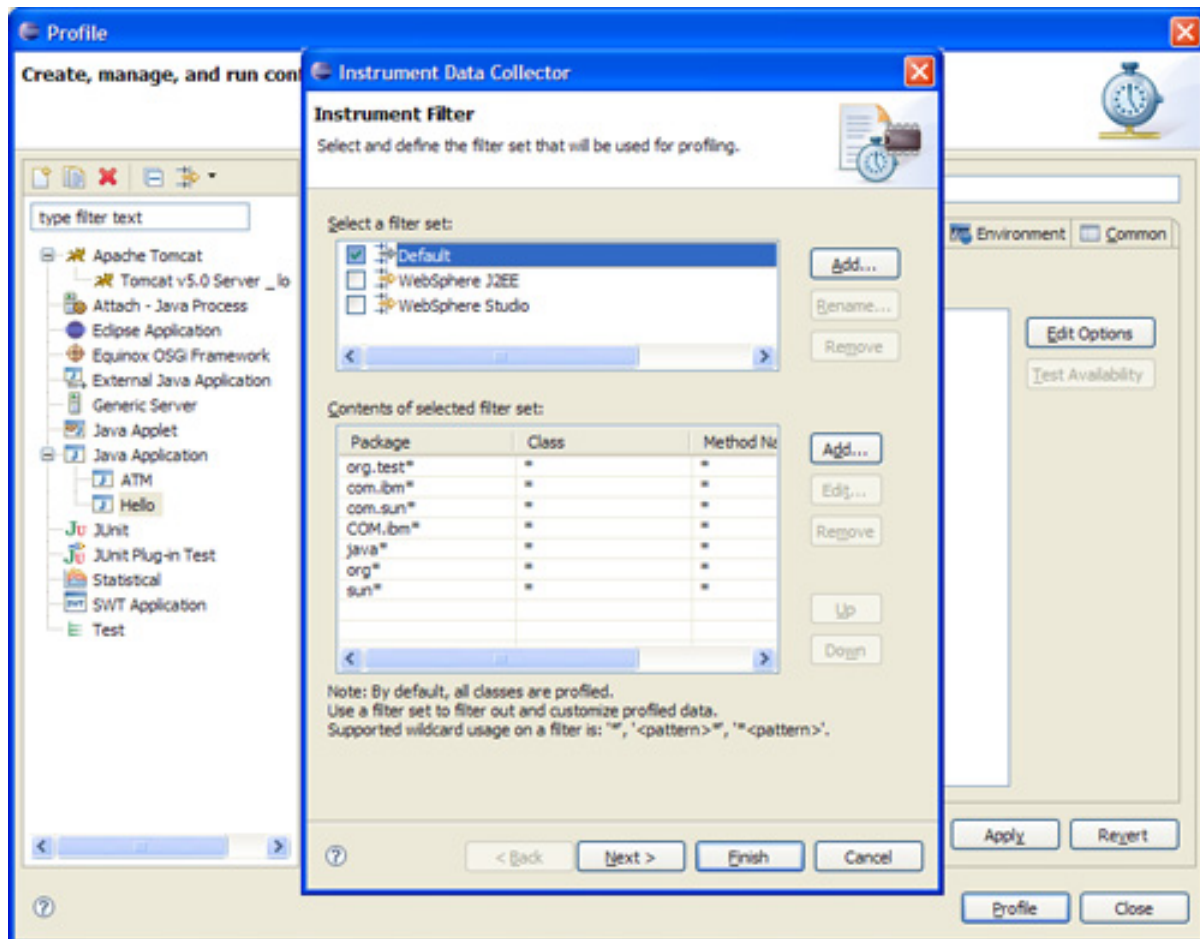


Figure 6. Click Edit Options on Instrument Collector to configure options



In the Instrument Collector, filters can be added or removed to help the user determine what packages, classes, or methods should be included or excluded during instrumentation. You must add a filter to include the package(s) in which your application resides.

Figure 7. Check off ARM analysis type

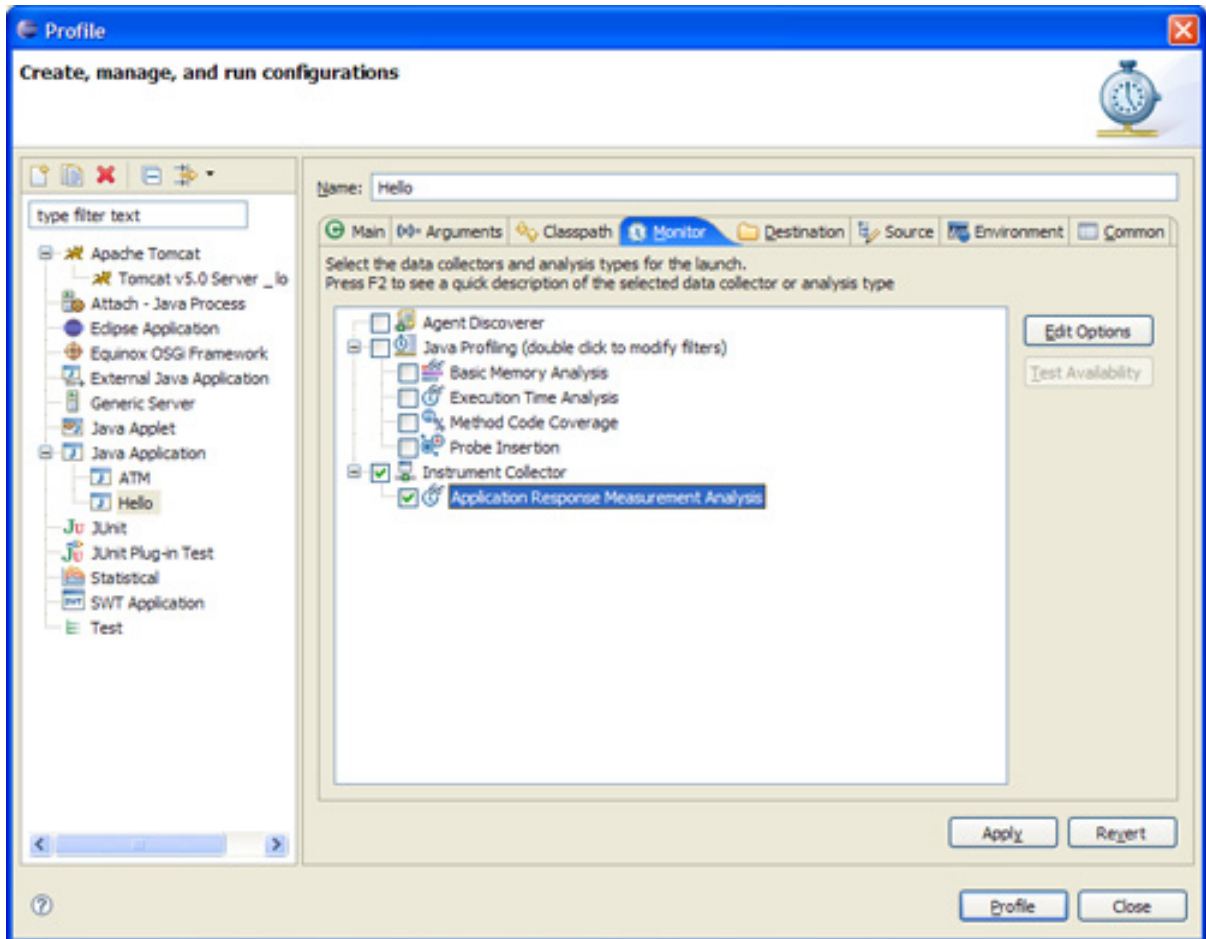
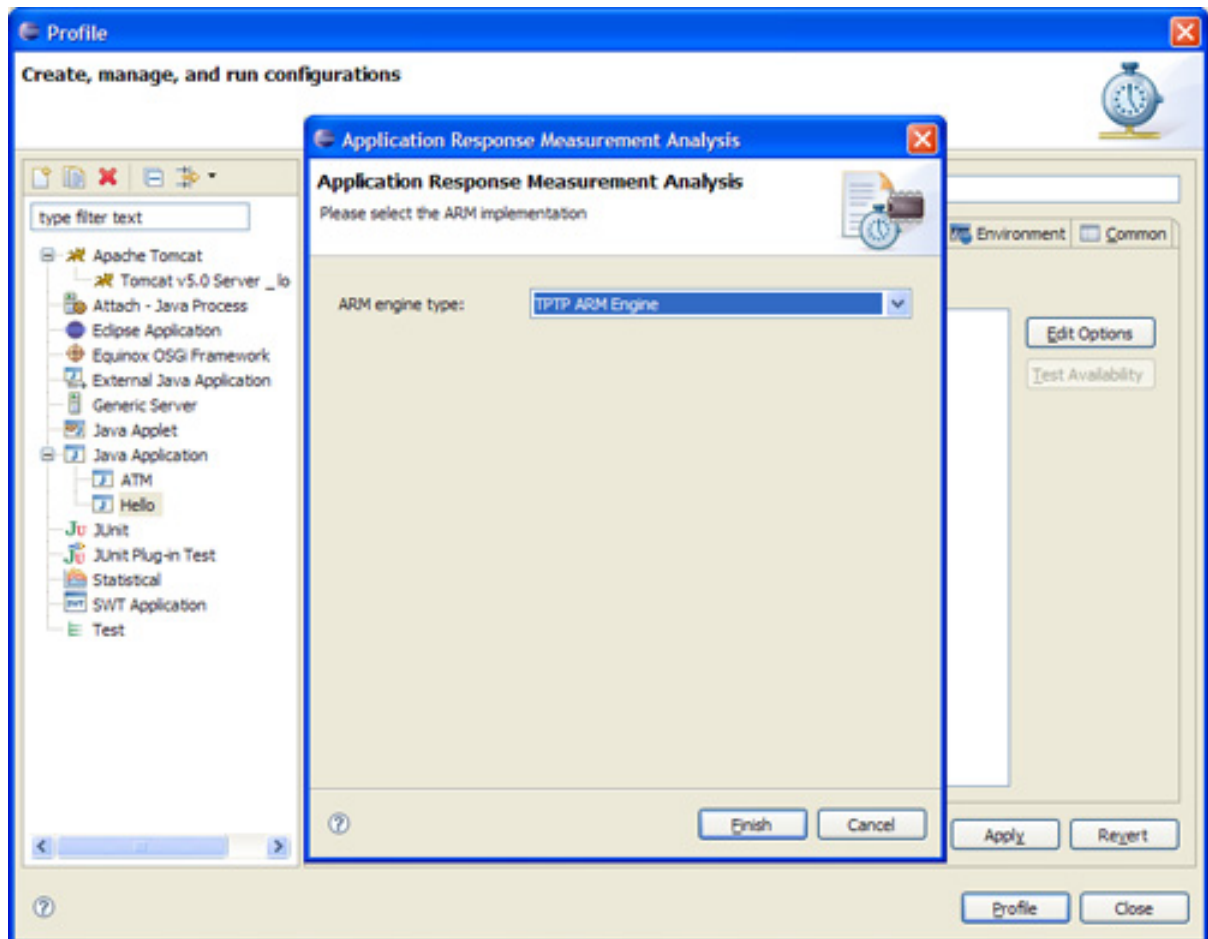


Figure 8. Click Edit Options on analysis type and pick TPTP ARM Engine (selected by default)



Then click **Profile** and run the application.

View and analyze collected ARM data

ARM data is stored in the TPTP trace data model, where it is available for viewing using any of the standard or custom TPTP views. The first two views shown here are from the HelloWorld application.

Figure 9. Execution Statistics view

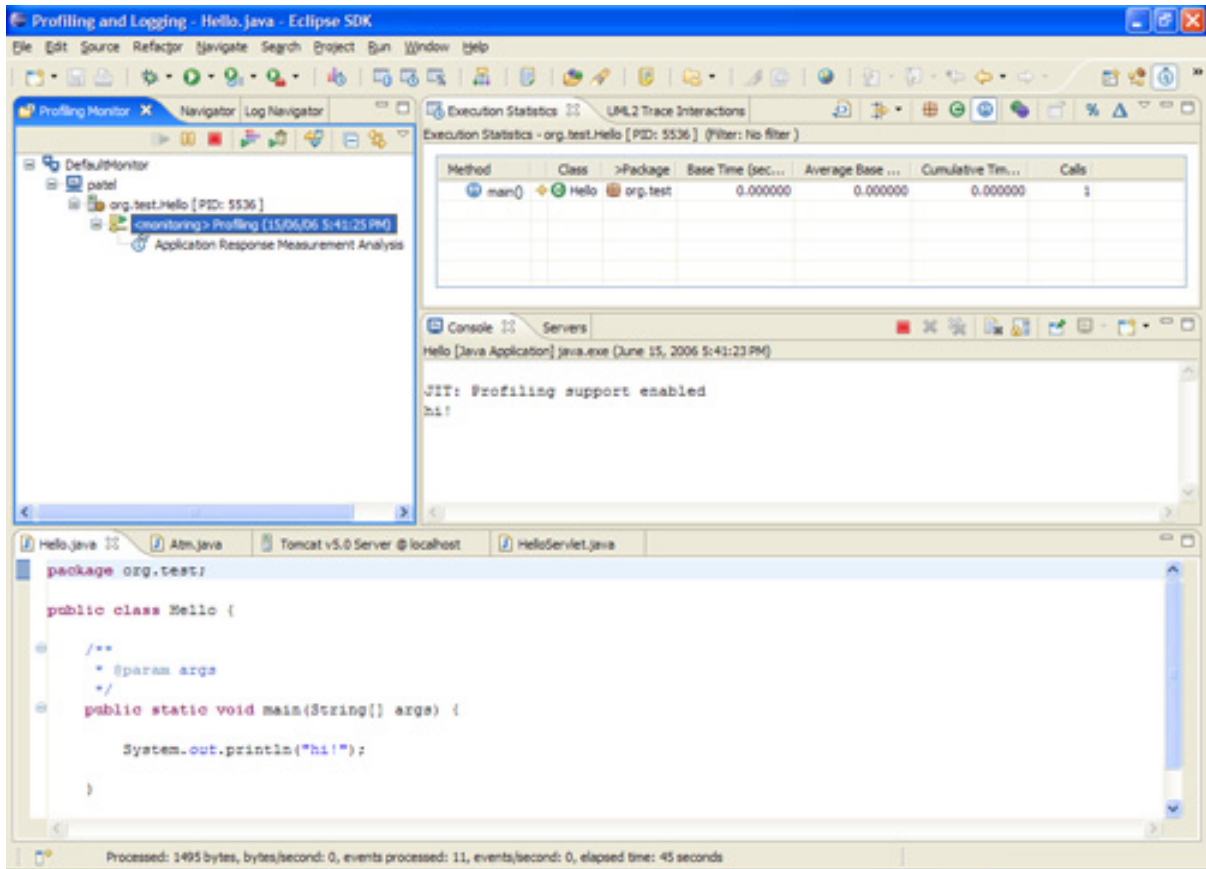
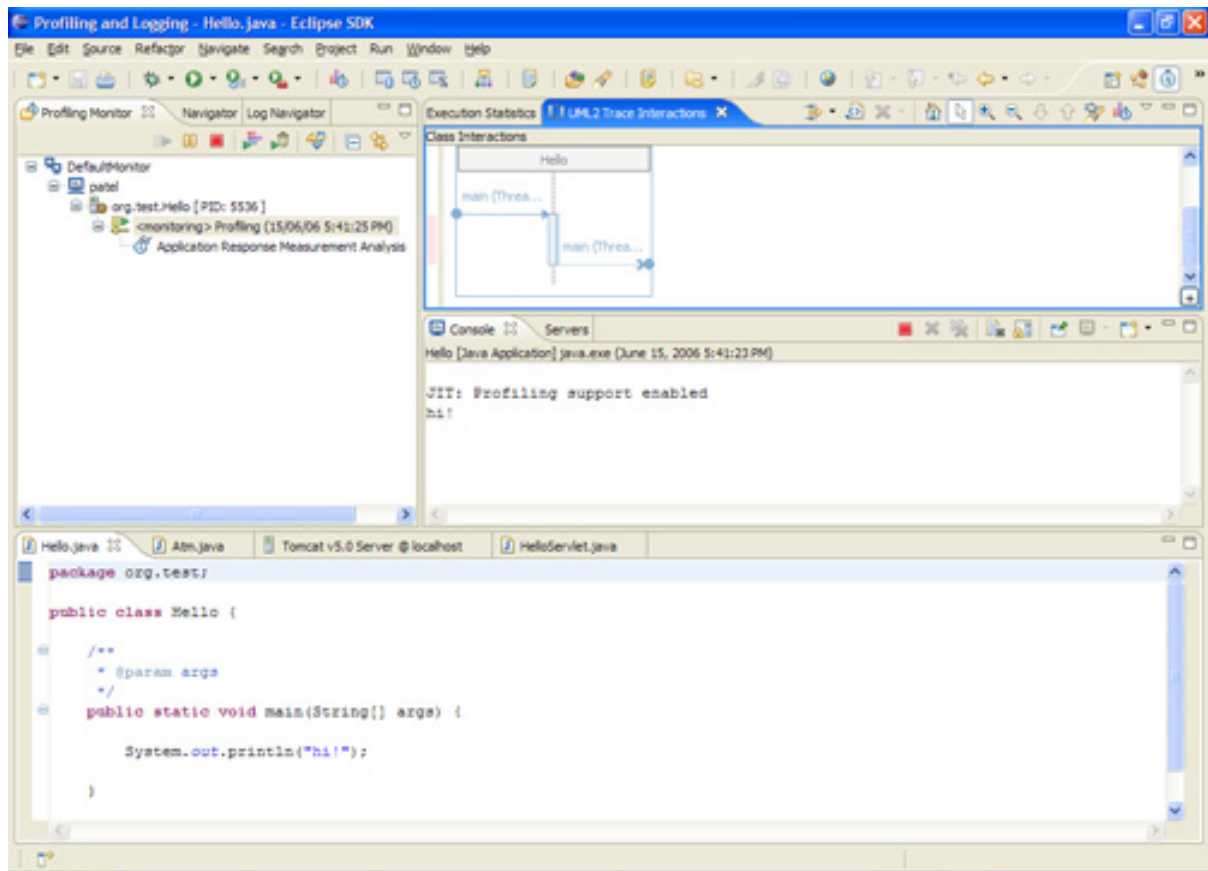


Figure 10. UML2 Trace Interaction view



The next three figures show views from a more complex application based on an automated teller machine (ATM). Other views standard in TPTP include Profiling Monitor, Code Coverage Statistics, and Execution Flow.

Figure 11. Execution Statistics view

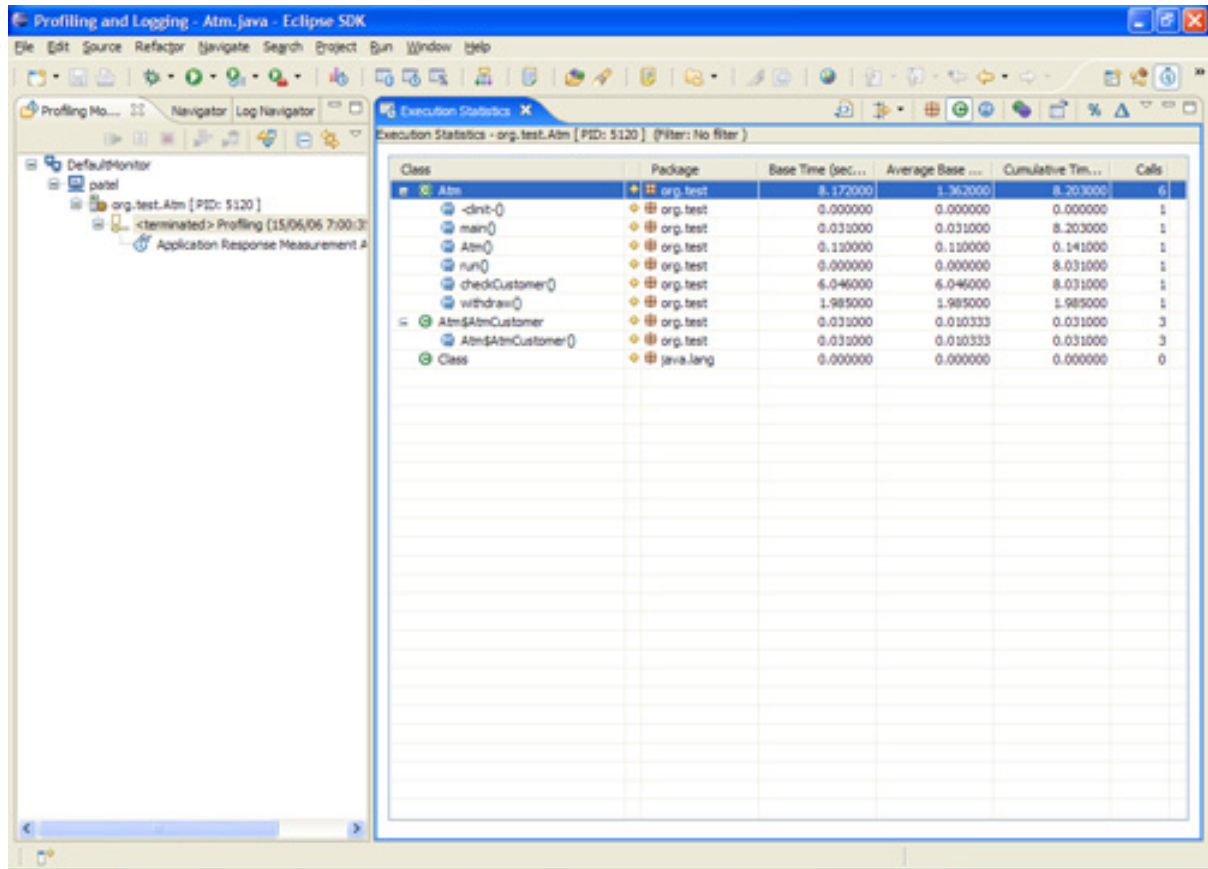


Figure 12. UML2 Trace Interaction view shows beginning of ATM transaction

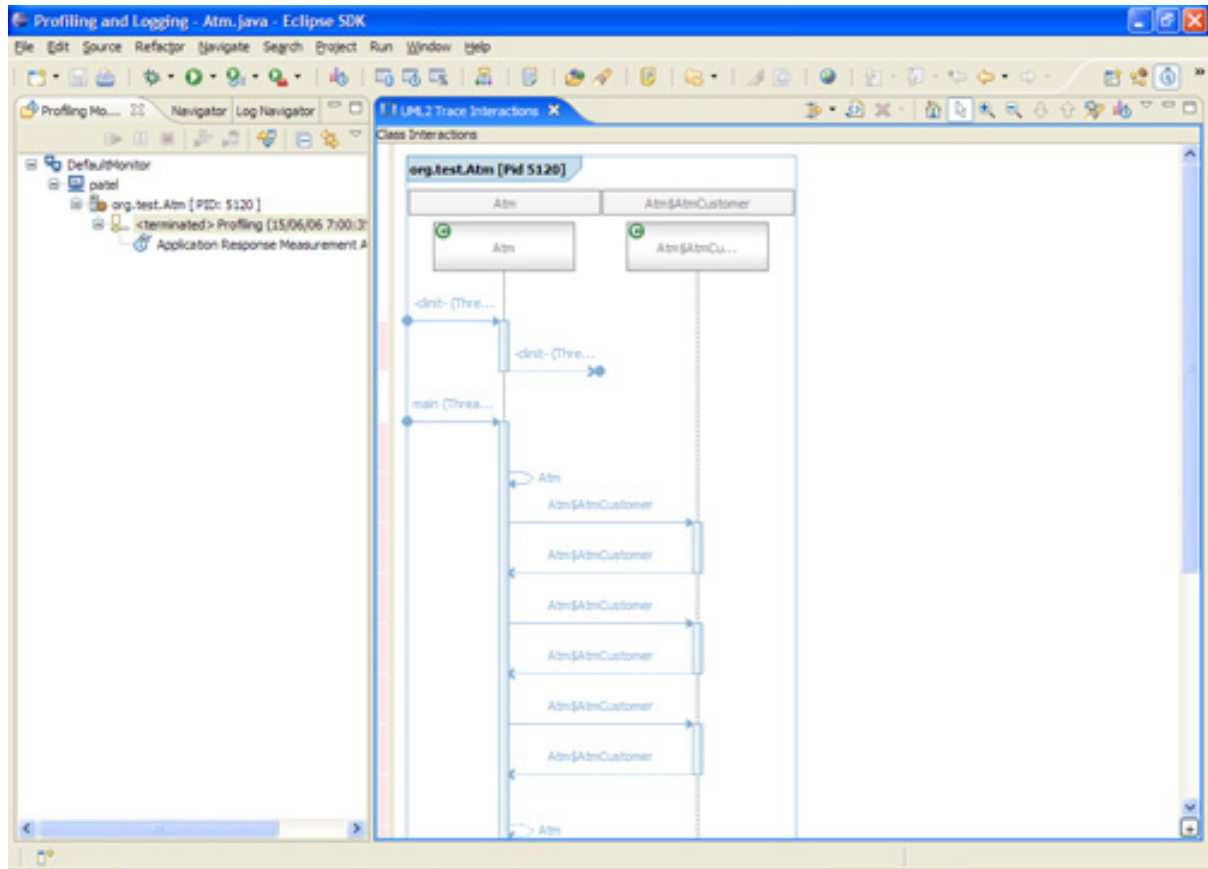


Figure 13. This UML2 Trace Interaction view shows actual bottleneck in the application

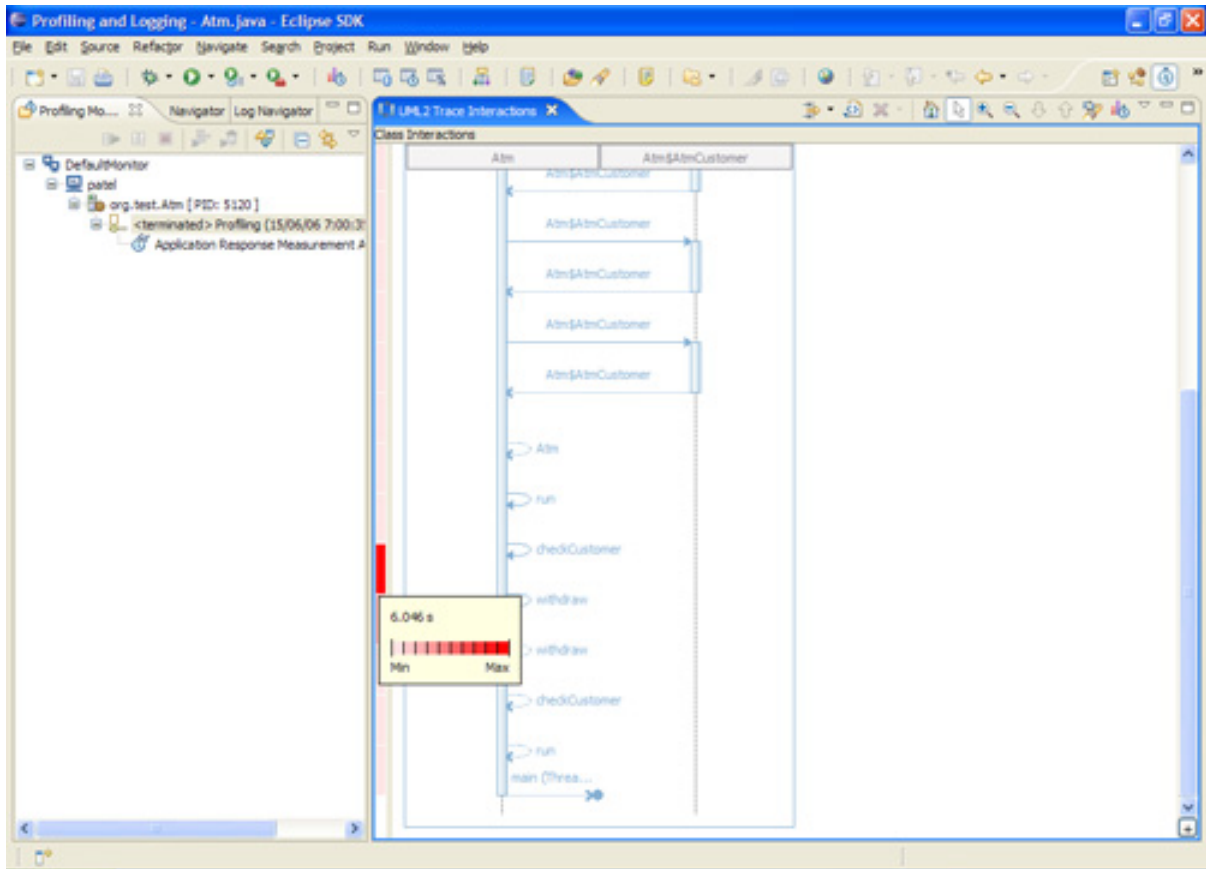
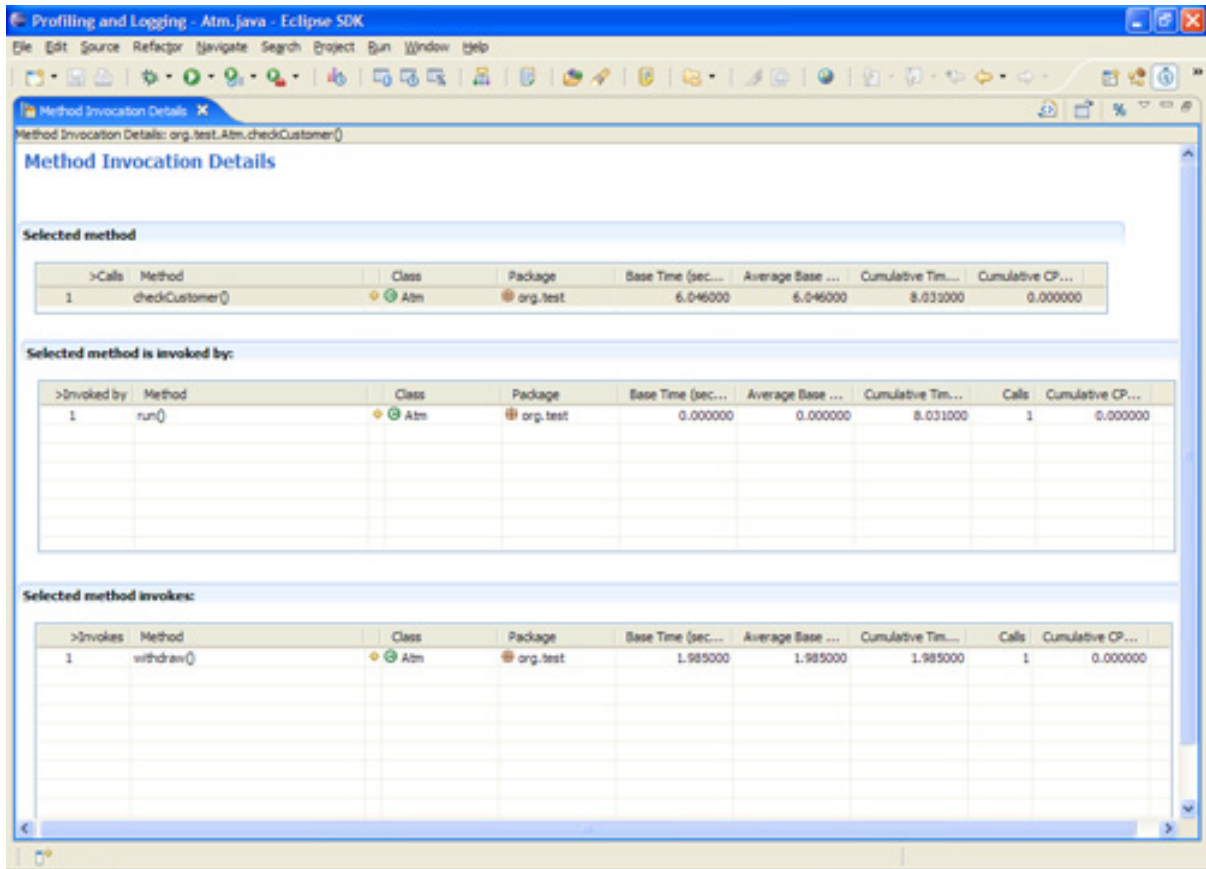


Figure 14. Method Invocation Details view



Section 8. Future direction

Current plans include adding enhanced types of instrumentation to the Probekit Library to support such technologies as Web services and native applications. We're looking to provide a more elaborate library for J2EE applications. The extended library could include servlet, Enterprise JavaBeans (EJBs), Remote Method Invocation (RMI), Java Database Connector (JDBC), and Java Web Services.

Other enhancements include support for instrumentation to work with the application servers supported by the Eclipse Web Tools Project (WTP), including IBM WebSphere®, Apache Tomcat, JBox, JOnAS, and Microsoft IIS. In addition, we look to capture the SQL statements executed during a business transaction on various databases, such as IBM DB2®, Cloudscape, and MySQL.

While Java and J2EE are leading technologies for application development, there are other technologies in widespread use, notably .NET (Visual Basic.NET or C#) and scripting languages (PHP, Perl, Ruby on Rails). Support for those technologies

could be added.

A particularly important issue is that the ARM V4.0 standard does not define the format of the correlator. This means that ARM implementations are not interoperable. As more ARM implementations arrive, we can expect this issue to start to gain prominence.

Resources

Learn

- Read "[CORBA Metaprogramming Mechanisms, Part 1](#)" on Dr. Dobb's Portal to learn the concepts and components associated with CORBA Portable Interceptors.
- For more information about Probekit, see the IBM [Probekit Reference](#).
- Learn about the [Eclipse Test & Performance Tools Platform \(TPTP\) Project](#).
- Check out [Application Response Measurement \(ARM\)](#) at OpenGroup.org.
- Find out more about [ARM technology](#) at OC Systems.
- Attend [EclipseCon](#), the premier Eclipse conference.
- For an excellent introduction to the Eclipse platform, see "[Getting started with the Eclipse Platform](#)."
- Expand your Eclipse skills by visiting IBM developerWorks [Eclipse project resources](#).
- developerWorks offers interesting [podcasts](#) for software developers.
- Visit the developerWorks [Open source zone](#) for extensive how-to information, tools, and project updates to help you develop with open source technologies and use them with IBM's products.

Get products and technologies

- Download [Eclipse Test & Performance Tools Platform \(TPTP\)](#).
- Download the [Eclipse Platform](#) and get started with Eclipse now.
- See the latest [Eclipse technology downloads](#) at IBM [alphaWorks](#).
- Innovate your next open source development project with [IBM trial software](#), available for download or on DVD.

Discuss

- The [Eclipse Platform newsgroups](#) should be your first stop to discuss questions regarding Eclipse. (Selecting this will launch your default Usenet news reader application and open eclipse.platform.)
- The [Eclipse newsgroups](#) has lots of resources for people interested in using and extending Eclipse.
- Get involved in the developerWorks community by participating in [developerWorks blogs](#).

About the authors

Ashish Patel



Ashish Patel is a software architect and development lead working at the IBM Toronto Software Lab in Canada. He has more than seven years of industry experience in software architecture and development, and four years of business development and entrepreneurial experience. He participated in IBM's premiere internship program, Extreme Blue, where he co-founded the IBM Performance Optimization Toolkit. He has worked with IBM for two years on numerous start-up products in the software development and test area. Prior to joining IBM, he created software solutions for one of the largest integrated petroleum companies in the world and one of Canada's top providers of energy. He also founded a privately held software development and consultancy firm, where he was appointed the president and chairman of the corporation, which he operated for four years. He also holds a computer engineering degree from the University of Alberta.

Oliver E. Cole



Oliver E. Cole is president of OC Systems Inc., a software company that develops, sells, and supports advanced software instrumentation tools. He has more than 25 years of extensive hands-on experience in developing, testing, and performance tuning large-scale mission-critical software for a variety of organizations, including commercial and international interests. Before founding OC Systems, Cole worked on a number of high-reliability real-time systems for the U.S. military. He is a member of the Eclipse Test & Performance Tools Platform (TPTP) Project Management Committee, speaks regularly at industry conferences, and has had a number of articles published in industry publications.