International Journal of Computing and Business Research (IJCBR)

ISSN (Online) : 2229-6166

Volume 2 Issue 1 2011

MODIFICATION OF SOFTWARE ON THE FLY

TIRLOK GABBA

A.P, Department of M.Tech (CSE)

Bhiwani Institute of Technology & Science, Bhiwani

E-mail : trilok_gaba@yahoo.in

ABSTRACT:

There are different phases of the software development life cycles (SDLC). Every phase requires different amount of time to complete. Coding Phase is most critical phase of SDLC. After implementation phase, it is very hard to do the modification or changes. Execution time is used to make the software reliable. Software gets obsolete change with the user's requirement change with time. So an approach is used to modify the software on-the -fly that is reliable in term of money and time. It's save lots of time and provides benefit economically. The Java Platform Debugger Architecture (JPDA) technology from Sun helps when you need to debug a running Java application in all situations. JPDA provide a interface designed for the use of the front end and backend. Basically, JPDA is collection of the API and protocols that helps with dubbing java code. The JPDA is multilayered architecture that breaks the debugging application into two parts: first part is being debugging and the second is interface. The dubbing part is known as backend. And with the second part the developers provides the interface or link to the newly made application (front end).The tools IBM's eclipse is used for the on the fly modification. It can modify locally and remotely. There is no need to stop the execution of the application while doing the modification. By using on-the-fly modification technique, developer can save execution time and make the software reliable.

Keywords: Java Debug Interface, Java Development Tools, Java Debug Wire Protocol, Java Platform Debugger Architecture, Java Virtual Machine, JVM Debug Interface, JVM Tool Interface, Virtual Machine.

INTRODUCTION:

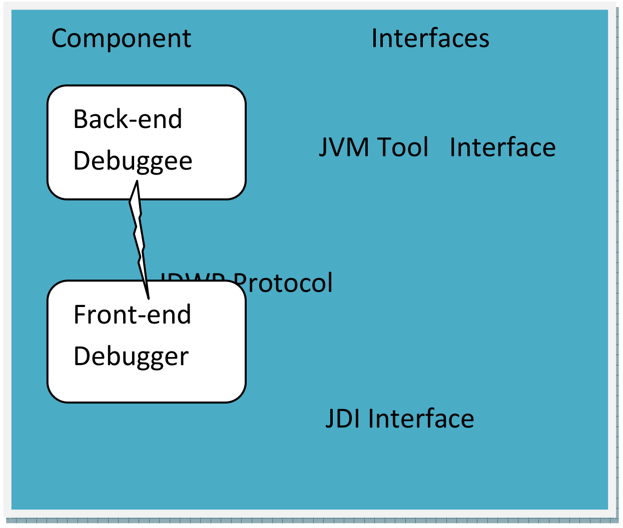
One characteristic that is constant in the software industry today is the “change”. Change is one of the most critical aspects of software development and management. New tools and approaches are announced almost every day. The impact of these developments is often very extensive and raises a number of issues that must be addressed by the software developers. Most important among them friendliness of software products. There may be occasions when you find bugs in a running application, fix it and want to test the fix without restarting the application. This is very common if you are working on enterprise level applications, which have less or absolutely no acceptable downtime. Many applications have to run continuously and cannot afford to be down for improvement and maintains. A common solution is to design the system in a special way and run programs on a specially configured, redundant hardware. An alternative method is investigated where parts of the application is updated at runtime.

JPDA is a collection of APIs aimed to help with debugging Java code. The JPDA set of tools is available in J2SE starting from the 1.2.2 version, and beginning from 1.3.x, it's included directly into the J2SE package.

It's important to understand that JPDA is not an application or a debugging tool but a set of well-designed and implemented interfaces and protocols. Sun's mission in this standard is to provide an infrastructure, so third-party tools and debuggers can use it in the most efficient way. There are a lot of excellent debuggers and IDEs that use JPDA, including such widely recognized tools as Borland JBuilder, Oracle Developers, IBM Eclipse, and many others. However, Sun provides a reference implementation in its traditional command-line debugger jdb, rewritten in Java 1.3 for supporting JPDA.

CONCEPT TO MODIFY ON-THE-FLY:

The Java Platform Debugger Architecture (JPDA) technology is a multitier architecture that allows you to debug Java applications in all situations easily. The JPDA consist of two type of interface and protocol Java Wired debugging Protocol (JDWP) and join the two ends front-end and back-end. It is not necessary to use the JPDA on the desktop application but it can also be used on the client server architecture. The JVM tool interface assures that Virtual Machine (VM) must be present on the debugging system. The Java Debug Wire Protocol (JDWP) describes the format of debugging information and requests transferred between the process being debugged and a debugger front end, which implements the JDI, as in Eclipse. The program being debugged is often called the debug gee. The JDI is a high-level interface to define the information and requests used for remote debugging. In establishing a connection between a debugger application and target VM, one side acts as a server and listens for a connection. At some later time, the other side attaches to the listener and establishes a connection. The connections allow the debugger application or the target VM to act as a server. The communications among processes can be running on one machine or different machines. But here have taken the local looping address 127.0.0.1 with specific port number to do the socket programming on the same system. Between the back-end and a front-end there is a communication channel working on top of the JDWP protocol; therefore, the debuggee and the debugger can be located on the same box or on different boxes.



The communications channel is the link between the front- and back-ends of the debugger. It is consisting of two mechanisms: a connector and a transport. A connector is a JDI object that is the means by which a connection is established between the front- and back-ends. There can be three types of connectors:

1. Listening: The front-end listens for an incoming connection from the back-end.
2. Attaching: The front-end attaches to an already running back-end.
3. Launching: The front-end actually launches the Java process that will run the debuggee code and the back-end.

A transport is the underlying mechanism used to move bits between the front-end and the back-end. The transport mechanism that must be used is not specified in the JPDA specification; possible mechanisms include: sockets, serial lines, and shared memory. However, the format and semantics of the serialized bit-stream flowing over the channel is specified by the JDWP. Most IDEs and debuggers support two transport types: shared memory (if the debuggee and debugger are located on the same box) and socket (the debuggee and debugger can be located anywhere, including the same box).

TESTBED FOR MODIFICATION:

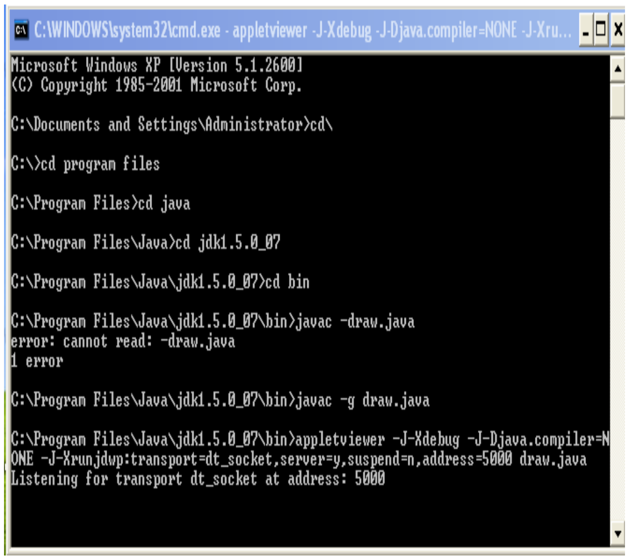
The work is implemented in the JDK1.3 or its greater version. The small piece of the code is written and compiles it in the debugging mode.

HYPOTHESIS:

H1: The correctness and completeness of modified application is verified by the programmer.

H2: The failure rate of the system is decreased when the fault is removed by on fly modification. And assume that reliability model consider that on fly modification correct one fault from the software at a time.

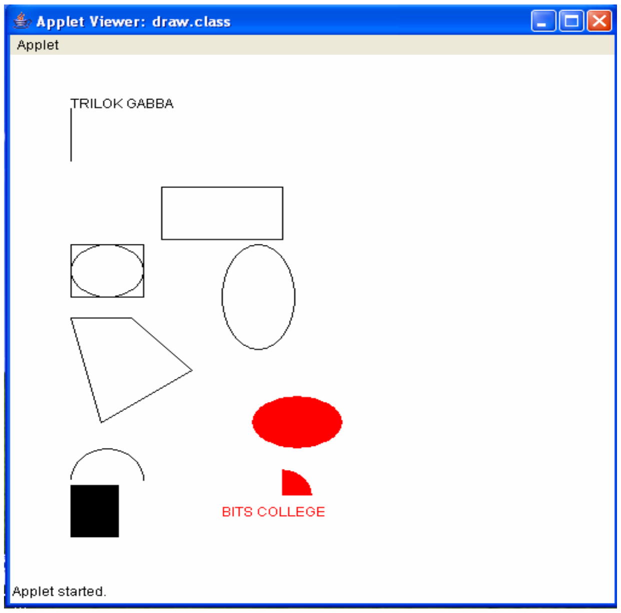
. Compile and debug the example with following command:



```
C:\WINDOWS\system32\cmd.exe - appletviewer -J-Xdebug -J-Djava.compiler=NONE -J-Xru...
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd\

C:\>cd program files

C:\Program Files>cd java

C:\Program Files\Java>cd jdk1.5.0_07

C:\Program Files\Java\jdk1.5.0_07>cd bin

C:\Program Files\Java\jdk1.5.0_07\bin>javac -draw.java
error: cannot read: -draw.java
1 error

C:\Program Files\Java\jdk1.5.0_07\bin>javac -g draw.java

C:\Program Files\Java\jdk1.5.0_07\bin>appletviewer -J-Xdebug -J-Djava.compiler=N
ONE -J-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5000 draw.java
Listening for transport dt_socket at address: 5000
```

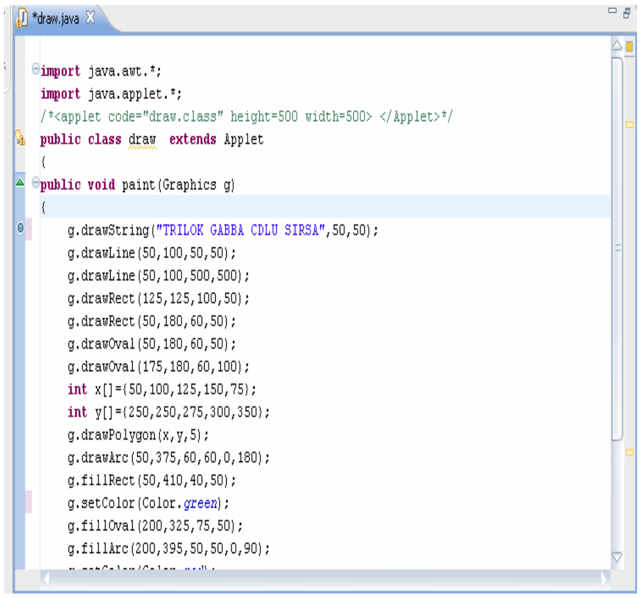
1. Java –g draw.java //for compile the applet

2.Appletviewer-j-Xdebug-j-Xrunjdwp: transport=dt_socket, server=y, suspend=n, address=5000



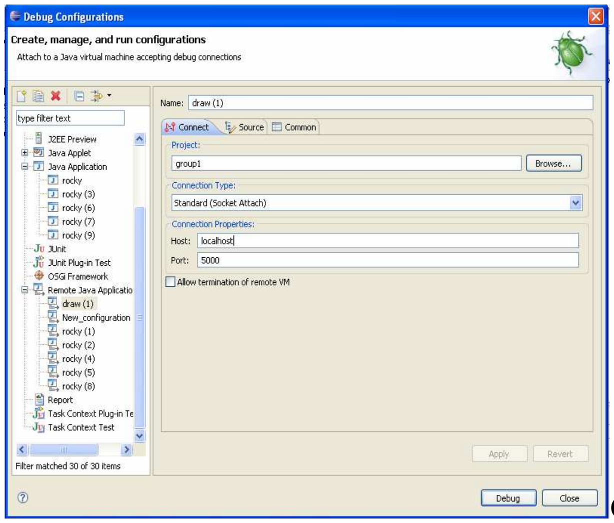
draw.java // for debugging the applet. **Result of the program before the modification dynamically**

Let us consider a simple java Applet program that is used for drawing the objects in the appletviewer window. The Application program is run in the debugging mode. It is executing this application with help JDWP protocol. There is concept of socket programming. The Socket is combination of the two IP address and the port no. The 32 bits IP address is used for identifying the remote machine. And the socket is listening for a particular port specified by the programmer. For the connection with remote java application, this tool provides the connection properties with host name and port. Host name is used to connect with the remotely java application. It can contain either the name of the host machine or the IP address of the machine. The specific port number is used for attaching with remotely java application.



```
import java.awt.*;
import java.applet.*;
/*<applet code="draw.class" height=500 width=500> </Applet>*/
public class draw  extends Applet
{
public void paint(Graphics g)
{
    g.drawString("TRILOK GABBA CDLU SIRSA",50,50);
    g.drawLine(50,100,50,50);
    g.drawLine(50,100,500,500);
    g.drawRect(125,125,100,50);
    g.drawRect(50,180,60,50);
    g.drawOval(50,180,60,50);
    g.drawOval(175,180,60,100);
    int x[]={50,100,125,150,75};
    int y[]={250,250,275,300,350};
    g.drawPolygon(x,y,5);
    g.drawArc(50,375,60,60,0,180);
    g.fillRect(50,410,40,50);
    g.setColor(Color.green);
    g.fillOval(200,325,75,50);
    g.fillArc(200,395,50,50,0,90);
```

The modified java code with same class name.



Create, manage, and run configurations
Attach to a Java virtual machine accepting debug connections

Name: draw (1)

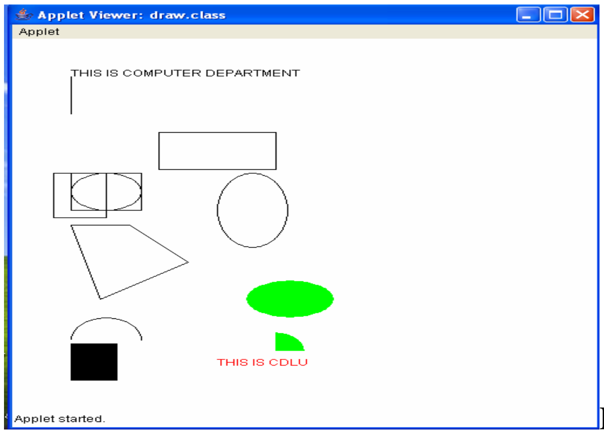
Project: group1

Connection Type: Standard (Socket Attach)

Connection Properties:
Host: localhost
Port: 5000

☐ Allow termination of remote VM

Configuration used for connecting remotely debugging application.



Result of the program after the modification dynamically.

With the help IBM's eclipse tool, the remotely application is modified on-the-fly. Because of changing it at run time, we have used the port address that is used by JVM for listening. The modified version of the program is given below. So here is opportunity for developer to save their major time while developing the application.

CONCLUSION:

The technique used here can make the software reliable by decreasing the numbers of faults occurs in the software or by modify the application. It enhances the reliability in terms of time by saving it. Failure can be removed in the application without stopping its execution. we assume that validation and verification of the modified application is done by the programmers .

REFERENCES:

[1] Reddy, R.K., "Hot Swapping with Java", Posted On June 20, 2007.

[2] Leszek, P."Debugging with the Eclipse Platform." posted on April 3, 2009.

[3] Users new to Eclipse should look at the Eclipse start here.

[4]I.Lee,"DYMOS:A Dynamic Modification System" PhD thesis ,University of Wisconsin,1983.

[5] Learn in-depth about Eclipse's Debug Framework through this Eclipse tutorial.

[6] Segal, M.E. and Frieder, O., “On-the-fly program modification: system for dynamic updating”, IEEE software, page53-65, March, 1993.

[7]Frieder, O. and Segal, M.E., “On dynamically updating a computer program: from concept to prototype”, J. System software, 14(2):111-128, Sept 1991.

[8] Fabry, R.S., “How to design systems in which module can be changed on the fly”, In Proc. 2nd Int. Conf Software Engg, 1976.