```
package com.ustb.bpel.bean;
import java.awt.Color;
import java.awt.Component;
import java.io.FileNotFoundException;
import java.io.IOException;
import org.apache.log4j.AppenderSkeleton;
import org.apache.log4j.Logger;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import javax.management.ObjectName;
import javax.swing.DefaultComboBoxModel;
import javax.swing.DefaultListModel;
import javax.swing.JLabel;
import com.ustb.bpel.constants.Constants;
import com.ustb.bpel.log.TomcatLog;
import com.ustb.bpel.xml.*;
import com.ustb.bpel.xml.impl.BpelFileImpl;
import com.ustb.bpel.xml.impl.ConfigFileImpl;
public class Facade {
	LogActionMonitor logActionMonitor;
	public void setLogActionMonitor(LogActionMonitor logActionMonitor) {
		this.logActionMonitor = logActionMonitor;
	}
	public String getLastBPELFilePath(){
	Configuration config=(Configuration)Constants.beanFactory.getBean("configfileImpl");
		 return config.getLastBPELFilePath();
	}
	public String getLastBuildFilePath(){
		 Configuration
	config=(Configuration)Constants.beanFactory.getBean("configfileImpl");
		 return config.getLastBuildFilePath();
	}
	public void setLastBPELFilePath(String path){
		Configuration config=(Configuration)Constants.beanFactory.getBean("configfileImpl");
		config.setLastBPELFilePath(path);
	}
	public void setLastBuildFilePath(String path){
		Configuration config=(Configuration)Constants.beanFactory.getBean("configfileImpl");
		config.setLastBuildFilePath(path);
	}
	public String getServerLogString() {
		return Constants.serverLogString;
	}
	public void startGetLogThread() throws InterruptedException {
		TomcatLog tomcatLog=new TomcatLog();
		try {
		tomcatLog.startTomcatlog();
		}
	catch (IOException e)
```

```
          {
                  e.printStackTrace();
             }
        }
        public String getTOMCAT_HOME(){
             Configuration config=(Configuration)Constants.beanFactory.getBean("configfileImpl");
             return config.getTOMCAT_HOME();
        }
        public void setTOMCAT_HOME(String path){
             Configuration config=(Configuration)Constants.beanFactory.getBean("configfileImpl");
             config.setTOMCAT_HOME(path);
        }
        public void processLog(String logString, DefaultListModel model) {
             String actionString = logActionMonitor.processLog(logString);
                  if (actionString.equals("deployed")||actionString.equals("undeployed")) {
                       System.out.println("model changed: deployed& undeployed");
}
        }
public void setEngineStateListener(String logString,JLabel label){
             if(logString.contains("ActiveBPEL In-Memory Configuration Started")){
             label.setText("completed");
             label.setForeground(Color.BLUE);
             }
        }
   public void setDeploymentListener(String logsString, JLabel label, JLabel label2){
             if(logsString.contains("deployment")){
                  label.setForeground(Color.BLUE);
                  label.setText("deploy success");
             }else if(logsString.contains("undeployment")){
                  label2.setForeground(Color.BLUE);
                  label2.setText("undeploy success");
             }
        }
        public void addAppender(AppenderSkeleton appender){
             Logger.getRootLogger().addAppender(appender);
        }
}
package alan.testcase.executor;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.xml.namespace.QName;
import javax.xml.rpc.ParameterMode;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
public class RunCase {
	private static Map<String, QName> typeMap = new HashMap<String, QName>();
	static {
		typeMap.put("string", org.apache.axis.Constants.XSD_STRING);
		typeMap.put("int", org.apache.axis.Constants.XSD_INT);
		typeMap.put("double", org.apache.axis.Constants.XSD_DOUBLE);
		typeMap.put("float", org.apache.axis.Constants.XSD_FLOAT);
	}
	List<String> paramNameList = new ArrayList<String>();
	List<String> paramTypeList = new ArrayList<String>();
	String returnType = "";
	public List<ExecutionResult> execute(String endpoint, String operationName,
			String testcaseFilePath, String MutantName) {
		List<ExecutionResult> executionResults = new ArrayList<ExecutionResult>();
		BufferedReader reader = null;
		BufferedWriter writer = null;
		try {
			String resultFilePath = testcaseFilePath.substring(0,
					testcaseFilePath.lastIndexOf(".")) +"_"+MutantName+ "_result.txt";
			reader = new BufferedReader(new FileReader(new File(
					testcaseFilePath)));
			writer = new BufferedWriter(
					new FileWriter(new File(resultFilePath)));
			String line = null;
			line = reader.readLine();
			String[] inputs = line.split("#");
			int inputParamNumber = Integer.parseInt(inputs[0]);
			System.out.println(line);
			for (int i = 0; i < inputParamNumber; i++) {
				String[] inputParam = inputs[i + 1].split(",");
				String paramName = inputParam[0];
				String paramType = inputParam[1];
				paramNameList.add(paramName);
				paramTypeList.add(paramType);
			}
			returnType = inputs[inputs.length - 1];
			String[] args = new String[inputParamNumber];
			int no = 1;
			while ((line = reader.readLine()) != null) {
				ExecutionResult er = new ExecutionResult();
				String[] params = line.split("#");
				for (int i = 0; i < inputParamNumber; i++) {
					args[i] = params[i];
				}
				String expectedResult = params[inputParamNumber];
				String CaseInput = line;
```

```
                    long beginTime = System.currentTimeMillis();
                    String executedResult = executeTestCase(args, endpoint,
                            operationName);
                    long endTime = System.currentTimeMillis();
                    long interval = endTime - beginTime;
                    String status = expectedResult.equals(executedResult) ? "SUCCESS"
                            : "FAILURE";
                    double n = 0, m = 0;
                    m = status.equals("SUCCESS") ? m++ : n++;
                    er.setNo(no);
                    er.setBeginTime(getDateTime(beginTime));
                    er.setEndTime(getDateTime(endTime));
                    er.setInput(CaseInput);
                    er.setInterval(interval);
                    er.setExpectedResult(expectedResult);
                    er.setExecutedResult(executedResult);
                    er.setStatus(status);
                    executionResults.add(er);
                    writer.write(no + "#" + getDateTime(beginTime) + "#"
                            + getDateTime(endTime) + "#" + interval + "#"
                            + expectedResult + "#" + executedResult + "#" + status);
                    writer.write("\r\n");
                    writer.flush();
                    no++;
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            } finally {
                try {
                    if (reader != null) {
                        reader.close();
                    }
                    if (writer != null) {
                        writer.close();
                    }
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
            return executionResults;
        }
        public String executeTestCase(String[] params, String endpoint,
                String operationName) throws Exception {
            Call call = createCall(endpoint, operationName);
            String result = null;
            try {
                result = (String) call.invoke(params);
            } catch (Exception e) {
                result = "unexpected exception seen: " + e.toString();
            }
```

```
			if (result == null) {
				result = "<null>";
			}
			return result;
		}
		protected Call createCall(String endpoint, String operationName)
				throws Exception {
			Service service = new Service();
			Call call = (Call) service.createCall();
			call.setTargetEndpointAddress(endpoint);
			call.setOperationName(operationName);
			for (int i = 0; i < paramNameList.size(); i++) {
				String paramName = paramNameList.get(i);
				String paramType = paramTypeList.get(i);
				call.addParameter(paramName, typeMap.get(paramType),
						ParameterMode.IN);
			}
			call.setReturnType(typeMap.get(returnType));
			return call;
		}
		private String getDateTime(long date) {
			Date now = new Date(date);
			DateFormat df = DateFormat.getDateTimeInstance();
			return df.format(now);
		}
	}
	package alan.testcase.executor;
	import java.io.BufferedReader;
	import java.io.IOException;
	import java.util.ArrayList;
	import java.util.List;
	import java.io.File;
	import java.io.FileReader;
	import org.apache.log4j.LogManager;
	import org.apache.log4j.Logger;
	import org.apache.log4j.PropertyConfigurator;
	import alan.testcase.executor.*;
	public class Executor {
		private static Logger logger = LogManager.getLogger(Executor.class.getName());
		public static List<ExecutionMatrix> execute(String originOutputPath,String
		testcaseFilePath){
			List<ExecutionMatrix> executionResults = new ArrayList<ExecutionMatrix>();
			BufferedReader reader0 = null;
			BufferedReader reader1 = null;
			ArrayList<String> station=new ArrayList<String>() ;
			double m=0,n=0;
			try{
			reader0 = new BufferedReader(new FileReader(new File(
					originOutputPath)));
			reader1= new BufferedReader(new FileReader(new File(
                        testcaseFilePath)));
                String line = null;
                String Status;
                ExecutionMatrix er = new ExecutionMatrix();
                while ((line = reader0.readLine()) != null) {
                        ++n;
                        String mutantPath=reader1.readLine();
                if(!(line.equals(mutantPath))){
                        ++m;
                        Status ="F";
                }else{
        Status ="T";
        }
                station.add(Status);
                }
                String finalStatus=m==0?"SURVIVE":"KILLED";
                String
        MutantName=testcaseFilePath.substring(testcaseFilePath.lastIndexOf("\\")+1,testcaseFilePat
        h.lastIndexOf("."));
                double FDR=m/n;
                er.setMutantName(MutantName);
                er.setFDR(FDR);
                er.setStatus(station);
                er.setFinalStatus(finalStatus);
                executionResults.add(er);
    logger.info(er.getMutantName()+"#"+er.getStatus()+"#"+er.getFDR()+"#"+er.getFinalStatus());
        }
        catch (Exception ex) {
                ex.printStackTrace();
                }
                try {
                        reader0.close();
                        reader1.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
return executionResults;
        }
        }
}
        package alan.testcase.executor;
public class ExecutionResult {
        private int no;
        private String beginTime;
        private String endTime;
        private String CaseInput;
        private long interval;
        private String expectedResult;
        private String executedResult;
        private String status;
```

```
	private String MutantName;
	private double FDR;
	public int getNo() {
		return no;
	}
	public void setNo(int no) {
		this.no = no;
	}
	public String getBeginTime() {
		return beginTime;
	}
	public void setBeginTime(String beginTime) {
		this.beginTime = beginTime;
	}
	public String getEndTime() {
		return endTime;
	}
	public void setEndTime(String endTime) {
		this.endTime = endTime;
	}
	public String getInput() {
		return CaseInput;
	}
	public void setInput(String CaseInput) {
		this.CaseInput = CaseInput;
	}
	public long getInterval() {
		return interval;
	}
	public void setInterval(long interval) {
		this.interval = interval;
	}
	public String getExpectedResult() {
		return expectedResult;
	}
	public void setExpectedResult(String expectedResult) {
		this.expectedResult = expectedResult;
	}
	public String getExecutedResult() {
		return executedResult;
	}
	public void setExecutedResult(String executedResult) {
		this.executedResult = executedResult;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
```

```
    public String getMutantName() {
        return MutantName;
    }
    public void setMutantName(String MutantName) {
        this.MutantName = MutantName;
    }
    public double getFDR() {
        return FDR;
    }
    public void setFDR(double FDR) {
        this.FDR = FDR;
    }
}
package alan.testcase.executor;
import java.util.ArrayList;
public class ExecutionMatrix {
    private ArrayList<String> status;
    private String MutantName;
    private long interval;
    private double FDR;
    private String finalStatus ;
    public String getMutantName() {
        return MutantName;
    }
    public void setMutantName(String mutantName) {
        MutantName = mutantName;
    }
    public long getInterval() {
        return interval;
    }
    public void setInterval(long interval) {
        this.interval = interval;
    }
    public ArrayList<String> getStatus() {
        return status;
    }
    public void setStatus(ArrayList<String> status) {
        this.status = status;
    }
    public double getFDR() {
        return FDR;
    }
    public void setFDR(double fDR) {
        FDR = fDR;
    }
    public String getFinalStatus() {
        return finalStatus;
    }
    public void setFinalStatus(String finalStatus) {
        this.finalStatus = finalStatus;
    }
}
package com.ustb.bpel.view;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.GroupLayout.Alignment;
import javax.swing.GroupLayout;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.border.LineBorder;
import java.awt.Color;
import java.awt.SystemColor;
public class AboutFrame extends javax.swing.JFrame {
    public AboutFrame() {
        initComponents();
    }
    public void start() {
        this.setVisible(true);
        this.setSize(800, 750);
        toCenter();
    }
      public void toCenter() {
            Toolkit kit = Toolkit.getDefaultToolkit();
            Dimension screenSize = kit.getScreenSize();
            int screenWidth = screenSize.width/2;
            int screenHeight = screenSize.height/2;
            int height = this.getHeight();
            int width = this.getWidth();
            setLocation(screenWidth-width/2, screenHeight-height/2);
    }
    private void initComponents() {
        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Help");
        String text = "<html><body> "
                + "* ISV: Replace a variable identifier by another of the same type and same scope. <br>"
                + "* CFA: Replace an activity by an exit activity. <br>"
                + "* CDE: Applies the decision coverage criteria to a WS-BPEL composition. <br>"
                + "* CDC: Applies the decision/condition coverage criteria to a WS-BPEL composition.<br>"
               + "* CCO: Applies the condition coverage criteria to a WS-BPEL composition. <br>"
                + "* EAA: Replaces an arithmetic operator (+,-,*,div,mod) by another of the same type. <br>"
                + "* EEU: Removes the unary minus operator from any expression. <br>"
                + "* ERR: Replaces a relational operator (<, >, <=, >=, =, !=) by another of the same type. <br>"
				+ "* ELL: Replaces a logical operator (and, or) by one of the same type. <br>"
				+ "* ECC: Replaces a path operator (/, //) by another of the same type.<br>"
				+ "* ECN: Incrementing or decrementing values in one unit, adding or removing one digit.<br>"
				+ "* EMD: Replaces a duration expression by 0 or by half of the value.<br>"
				+ "* EMF: Replaces a deadline expression by 0 or by half of the value.<br>"
				+ "* EIU: Insert the XPath unary minus operator in aritmethic expressions. <br>"
				+ "* EIN: Insert the XPath negation function (not) in logic expressions. <br>"
				+ "* EAP: Replace a subexpression by its positive absolute value. <br>"
				+ "* EAN: Replace a subexpression by its negative absolute value. <br>"
				+ "* ACI: Changes the value of the createInstance attribute from 'yes' to 'no'.<br>"
				+ "* AFP: Modifying the value of the parallel attribute from 'no' to 'yes'.<br>"
				+ "* ASF: Replaces a sequence by a flow activity.<br>"
				+ "* AIS: Changes the isolated attribute of a scope to 'no'. <br>"
				+ "* AEL: Remove an activity.<br>"
				+ "* AIE: Removes an elseif element or the else element of an if activity. <br>"
				+ "* AWR: Replaces a while activity by a repeatUntil one and vice versa. <br>"
				+ "* AJC: Removes the joinCondition element. <br>"
				+ "* ASI: Exchanges two activities within a sequence activity. <br>"
				+ "* APM: Removes an onMessage element of a pick activity if there is more than one. <br>"
				+ "* APA: Removes the onAlarm element of a pick activity or an event handler.<br>"
				+ "* XMF: Removes one catch or the catchall construct of a fault handler. <br>"
				+ "* XMC: Removes a compensation handler definition. <br>"
				+ "* XMT: Removes a termination handler definition. <br>"
				+ "* XTF: Replaces the fault name by another one in the same scope. <br>"
				+ "* XER: Removes a rethrow activity. <br>"
				+ "* XEE: Removes an onEvent element. <br>"
				+ "<body></html>";
		JPanel panel = new JPanel();
		panel.setBorder(new LineBorder(SystemColor.activeCaption, 2, true));
				jLabel1 = new javax.swing.JLabel();
						jLabel1.setFont(new java.awt.Font("Consolas", 0, 22));
						jLabel1.setText("Mutation operators introduction:");
		javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
getContentPane());
		layout.setHorizontalGroup(
			layout.createParallelGroup(Alignment.LEADING)
				.addGroup(layout.createSequentialGroup()
					.addContainerGap()
					.addGroup(layout.createParallelGroup(Alignment.LEADING)
						.addComponent(panel, GroupLayout.PREFERRED_SIZE, 706, GroupLayout.PREFERRED_SIZE)
						.addComponent(jLabel1))
```

```
                        .addContainerGap(GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jLabel1)
                    .addPreferredGap(ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(panel,    GroupLayout.PREFERRED_SIZE,    631,
    GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())
        );
        panel.setLayout(null);
        jLabel2 = new javax.swing.JLabel();
        jLabel2.setBounds(10, 0, 688, 621);
        panel.add(jLabel2);
        jLabel2.setFont(new Font("Consolas", 0, 14));
        jLabel2.setText(text);
        getContentPane().setLayout(layout);
        pack();
    } private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
}
package com.ustb.bpel.view;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.BevelBorder;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.DefaultListModel;
import javax.swing.JFileChooser;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JList;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.File;
import java.util.List;
import javax.swing.JScrollPane;
import javax.swing.border.LineBorder;
import java.awt.Color;
import javax.swing.border.TitledBorder;
import java.awt.Font;
import javax.swing.ImageIcon;
import alan.testcase.executor.ExecutionMatrix;
import alan.testcase.executor.ExecutionResult;
import alan.testcase.executor.Executor;
import javax.swing.JComboBox;
import org.apache.log4j.LogManager;
import org.apache.log4j.Logger;
import javax.swing.JSeparator;
import javax.swing.UIManager;
public class ComparePanel extends JPanel
{
	private JTextField textField;
	private JList fileList = null;
	private DefaultListModel listModel = null;
	private JTable resultTable = null;
	private DefaultTableModel model = null;
	private JTable table;
	double ms=0,mq=0;
	private JTextField textField_1;
	private JTextField textField_2;
	private JTextField textField_3;
	public ComparePanel()
	{
		initLayout();
	}
	public void initLayout(){
	setLayout(null);
	listModel = new DefaultListModel();
	model = new DefaultTableModel();
	model.addColumn("No");
	model.addColumn("Mutant Name");
	model.addColumn("FDR");
	model.addColumn("Status");
	model.addColumn("FinalStatus");
	JPanel panel_1 = new JPanel();
	panel_1.setBorder(new
		TitledBorder(UIManager.getBorder("TitledBorder.border"),	"Analysis	Result",
	TitledBorder.CENTER, TitledBorder.TOP, null, null));
	panel_1.setBounds(22, 10, 854, 170);
	add(panel_1);
	panel_1.setLayout(null);
	JLabel lblNewLabel = new JLabel("Original Output:\r\n");
	lblNewLabel.setFont(new Font("Dialog", Font.PLAIN, 12));
	lblNewLabel.setBounds(10, 26, 123, 16);
	panel_1.add(lblNewLabel);
	textField = new JTextField();
	textField.setFont(new Font("Dialog", Font.PLAIN, 12));
	textField.setBounds(129, 19, 442, 28);
	panel_1.add(textField);
	textField.setColumns(10);
	JButton btnBrowse = new JButton("Browse");
	btnBrowse.setFont(new Font("Dialog", Font.PLAIN, 12));
	btnBrowse.setBounds(650, 12, 168, 29);
	btnBrowse.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				JFileChooser fileChooser = new JFileChooser();
				String startPath = System.getProperty("user.dir");
				fileChooser.setCurrentDirectory(new File(startPath));
				fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
				FileNameExtensionFilter filter = new FileNameExtensionFilter("txt Files", "txt");
				fileChooser.setFileFilter(filter);
				int index = fileChooser.showDialog(null, "Open File");
				if (index == JFileChooser.APPROVE_OPTION)
				{
					textField.setText(fileChooser.getSelectedFile().getAbsolutePath());
				}
			}
		});
		panel_1.add(btnBrowse);
		JButton btnSelectMutantOutput = new JButton("Select Mutant Output");
		btnSelectMutantOutput.setFont(new Font("Dialog", Font.PLAIN, 12));
		btnSelectMutantOutput.setBounds(650, 51, 168, 29);
		panel_1.add(btnSelectMutantOutput);
		JButton btnRun = new JButton("Run");
		btnRun.setFont(new Font("Dialog", Font.PLAIN, 12));
		btnRun.setBounds(650, 90, 168, 29);
		panel_1.add(btnRun);
		JButton btnReset = new JButton("Reset");
		btnReset.setFont(new Font("Dialog", Font.PLAIN, 12));
		btnReset.setBounds(650, 129, 168, 29);
		panel_1.add(btnReset);
		JScrollPane scrollPane_1 = new JScrollPane();
		scrollPane_1.setBounds(129, 57, 442, 101);
		panel_1.add(scrollPane_1);
		JList fileList_1 = new JList(listModel);
		fileList_1.setFont(new Font("Dialog", Font.PLAIN, 12));
		scrollPane_1.setViewportView(fileList_1);
		fileList_1.setBorder(null);
		JLabel lblMutantOutput = new JLabel("Mutant Output:\r\n");
		lblMutantOutput.setFont(new Font("Dialog", Font.PLAIN, 12));
		lblMutantOutput.setBounds(10, 105, 123, 16);
		panel_1.add(lblMutantOutput);
		JPanel panel = new JPanel();
		panel.setBorder(new TitledBorder(UIManager.getBorder("TitledBorder.border"), "Result",
		TitledBorder.CENTER, TitledBorder.TOP, null, null));
		panel.setBounds(561, 190, 315, 220);
		add(panel);
		panel.setLayout(null);
		JLabel lblNewLabel_1 = new JLabel("Number of Killed Mutant:\r\n");
		lblNewLabel_1.setFont(new Font("Dialog", Font.PLAIN, 12));
		lblNewLabel_1.setBounds(34, 54, 184, 15);
		panel.add(lblNewLabel_1);
		textField_1 = new JTextField();
		textField_1.setEditable(false);
```

```
textField_1.setBounds(185, 51, 66, 21);
panel.add(textField_1);
textField_1.setColumns(10);
textField_2 = new JTextField();
textField_2.setEditable(false);
textField_2.setColumns(10);
textField_2.setBounds(185, 103, 66, 21);
panel.add(textField_2);
JLabel lblTotalMutantNumber = new JLabel("Total Number of Mutant:\r\n");
lblTotalMutantNumber.setFont(new Font("Dialog", Font.PLAIN, 12));
lblTotalMutantNumber.setBounds(34, 106, 184, 15);
panel.add(lblTotalMutantNumber);
textField_3 = new JTextField();
textField_3.setEditable(false);
textField_3.setColumns(10);
textField_3.setBounds(185, 159, 66, 21);
panel.add(textField_3);
JLabel lblMutantScore = new JLabel("Mutant Score:\r\n");
lblMutantScore.setFont(new Font("Dialog", Font.PLAIN, 12));
lblMutantScore.setBounds(34, 162, 184, 15);
panel.add(lblMutantScore);
JPanel panel_2 = new JPanel();
panel_2.setBorder(new TitledBorder(null, "Output", TitledBorder.CENTER,
TitledBorder.TOP, null, null));
panel_2.setBounds(22, 190, 529, 220);
add(panel_2);
panel_2.setLayout(null);
JTable resultTable_1 = new JTable(model);
resultTable_1.setColumnSelectionAllowed(true);
resultTable_1.setBounds(346, 253, 204, 190);
resultTable_1.setShowHorizontalLines (true);
resultTable_1.setShowVerticalLines (true);
add(resultTable_1);
JScrollPane scrollPane_2 = new JScrollPane(resultTable_1);
scrollPane_2.setBounds(10, 20, 509, 190);
panel_2.add(scrollPane_2);
btnReset.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        textField.setText(null);
        model.setRowCount(0);
        listModel.removeAllElements();
        textField_3.setText(null);
        textField_1.setText(null);
        textField_2.setText(null);
    }
});
btnRun.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String originOutputPath = textField.getText();
        for(int i = 0; i < listModel.getSize(); i++) {
```

```
                    File file = (File)listModel.get(i);
                    String testcaseFilePath = file.getAbsolutePath();
                    System.out.println(testcaseFilePath);
            compareTestCase(originOutputPath,testcaseFilePath,i+1);
                }
            }
        });
        btnSelectMutantOutput.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
            JFileChooser fileChooser = new JFileChooser();
                fileChooser.setMultiSelectionEnabled(true);
                String startPath = System.getProperty("user.dir");
                fileChooser.setCurrentDirectory(new File(startPath));
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                FileNameExtensionFilter filter = new FileNameExtensionFilter("txt Files", "txt");
                fileChooser.setFileFilter(filter);
            int index = fileChooser.showDialog(null, "Open File");
                if (index == JFileChooser.APPROVE_OPTION) {
                    File[] files = fileChooser.getSelectedFiles();
                    for(int i = 0; i < files.length; i++) {
                        listModel.addElement(files[i]);
                    }
                }
            }
        });
        }
    private void compareTestCase(String originOutputPath,String testcaseFilePath, int i) {
            List<ExecutionMatrix>    executionResults    =    Executor.execute(originOutputPath,
        testcaseFilePath);
        Object[] rowData = new Object[5];
        for(int j = 0; j < executionResults.size(); j++) {
             ExecutionMatrix er = executionResults.get(j);
                    rowData[0] = i;
                    rowData[1] = er.getMutantName();
                    rowData[2] = er.getFDR();
                    rowData[3] = er.getStatus();
                    rowData[4] = er.getFinalStatus();
            System.out.println(i+"#"+er.getMutantName()+"#"+er.getFDR()+"#"+er.getStatus()+"#"
        +er.getFinalStatus());
                    ms=rowData[4]=="KILLED"?ms+1:ms+0;
                    mq++;
        java.text.DecimalFormat    df    =new    java.text.DecimalFormat("0.00");
        String st=df.format(ms/mq);
            textField_3.setText(" "+st);
          textField_1.setText(" "+ms);
          textField_2.setText(" "+mq);
        model.insertRow(model.getRowCount(), rowData);
        }
    }
    }
```

```
package com.ustb.bpel.view;
import javax.swing.JFileChooser;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextArea;
import javax.swing.JButton;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.util.List;
import javax.swing.JTextField;
import java.awt.Color;
import javax.swing.border.TitledBorder;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JSeparator;
import javax.swing.DefaultListModel;
import javax.swing.JScrollPane;
import javax.swing.JList;
import javax.swing.JTable;
import alan.program.Configuration.WSDLParser;
import alan.testcase.executor.*;
import javax.swing.border.LineBorder;
import java.awt.SystemColor;
import javax.swing.border.BevelBorder;
import javax.swing.UIManager;
public class RunPanel extends JPanel {
	String bpelFilePath;
	String bpelfile;
	private JTextField textField;
	JTextArea textArea_1;
	private DefaultListModel listModel = null;
	private DefaultListModel listModel2 = null;
	private DefaultTableModel model = null;
	String endpoint;
	String operationName;
	private static Process process;
	private static BufferedReader br;
	public RunPanel() {
		setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null, null));
		initLayout();
		initControl();
	}
	private void initControl() {
	}
	private void initLayout() {
        setLayout(null);
        listModel = new DefaultListModel();
        listModel2 = new DefaultListModel();
        model = new DefaultTableModel();
        model.addColumn("No");
        model.addColumn("Begin Time");
        model.addColumn("End Time");
        model.addColumn("Input");
        model.addColumn("Expected Result");
        model.addColumn("Excuted Result");
        model.addColumn("Status");
        JSeparator separator = new JSeparator();
        separator.setBounds(0, 0, 2, 418);
        add(separator);
        separator.setOrientation(SwingConstants.VERTICAL);
        separator.setForeground(Color.WHITE);
        JPanel panel_1 = new JPanel();
        panel_1.setBorder(new TitledBorder(null, "Run Test Case", TitledBorder.CENTER,
TitledBorder.TOP, null, null));
        panel_1.setBounds(12, 6, 869, 412);
        add(panel_1);
        panel_1.setLayout(null);
        JButton btnNewButton_1 = new JButton("Browse");
        btnNewButton_1.setBounds(646, 23, 145, 23);
        panel_1.add(btnNewButton_1);
        btnNewButton_1.setFont(new Font("Dialog", Font.PLAIN, 12));
        JLabel lblBpelFile = new JLabel("BPEL Build File:");
        lblBpelFile.setBounds(31, 25, 138, 19);
        panel_1.add(lblBpelFile);
        lblBpelFile.setFont(new Font("Dialog", Font.PLAIN, 12));
        textField = new JTextField();
        textField.setBounds(179, 25, 429, 23);
        panel_1.add(textField);
        textField.setColumns(10);
        JScrollPane scrollPane_3 = new JScrollPane();
        scrollPane_3.setBounds(179, 79, 429, 121);
        panel_1.add(scrollPane_3);
        JList list1 = new JList(listModel2);
        scrollPane_3.setViewportView(list1);
        JButton btnSelectMutant = new JButton("Select Mutant Files");
        btnSelectMutant.setBounds(646, 78, 145, 29);
        panel_1.add(btnSelectMutant);
        btnSelectMutant.addActionListener(new selectMutantAction());
        btnSelectMutant.setFont(new Font("Dialog", Font.PLAIN, 12));
        JLabel lblBpelMutantFiles = new JLabel("BPEL Mutant Files:");
        lblBpelMutantFiles.setBounds(31, 63, 138, 19);
        panel_1.add(lblBpelMutantFiles);
        lblBpelMutantFiles.setFont(new Font("Dialog", Font.PLAIN, 12));
                JScrollPane scrollPane_1 = new JScrollPane();
                scrollPane_1.setBounds(179, 224, 429, 131);
```

```
				panel_1.add(scrollPane_1);
				JList list = new JList(listModel);
				scrollPane_1.setViewportView(list);
					JButton btnSelectTestcase = new JButton("Select Test Cases");
					btnSelectTestcase.setBounds(649, 221, 142, 29);
					panel_1.add(btnSelectTestcase);
					btnSelectTestcase.setFont(new Font("Dialog", Font.PLAIN, 12));
					JLabel lblTestcaseFiles = new JLabel("Test Case Files:");
					lblTestcaseFiles.setFont(new Font("Dialog", Font.PLAIN, 12));
					lblTestcaseFiles.setBounds(27, 219, 111, 19);
					panel_1.add(lblTestcaseFiles);
					JButton btnDeployProcess = new JButton("Run Test Cases");
						btnDeployProcess.setFont(new                    Font("Dialog",
Font.PLAIN, 12));
						btnDeployProcess.setBounds(646, 281, 145, 23);
						panel_1.add(btnDeployProcess);
						JButton btnClear = new JButton("Clear\r\n");
						btnClear.setFont(new Font("Dialog", Font.PLAIN, 12));
						btnClear.setBounds(646, 332, 145, 23);
						btnClear.addActionListener(new ActionListener() {
							public void actionPerformed(ActionEvent e) {
							listModel2.removeAllElements();
							listModel.removeAllElements();
							textField.setText(null);
							}
						});
						panel_1.add(btnClear);
						btnNewButton_1.addActionListener(new
selectOriginalBPELAction());
						btnDeployProcess.addActionListener(new
DeployAction());
					btnSelectTestcase.addActionListener(new SelectTestcaseAction());
			}
	private class selectOriginalBPELAction implements ActionListener {
		public void actionPerformed(ActionEvent e) {
			JFileChooser jChooser = new JFileChooser();
			String startPath = System.getProperty("user.dir");
			jChooser.setCurrentDirectory(new File(startPath));
			jChooser.setDialogTitle("please select buile.xml file");
			int index = jChooser.showDialog(null, "Open File");
			if (index == JFileChooser.APPROVE_OPTION) {
				bpelFilePath = jChooser.getSelectedFile().getAbsolutePath();
				textField.setText(bpelFilePath);
				System.out.println(bpelFilePath);
			}
			WSDLParser.RunConfiguration(bpelFilePath);
			operationName = WSDLParser.configuration.getOperationName();
			endpoint = WSDLParser.configuration.getEndPoint();
		}
	}
```

```
private class SelectTestcaseAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setMultiSelectionEnabled(true);
                String startPath = System.getProperty("user.dir");
                fileChooser.setCurrentDirectory(new File(startPath));
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                FileNameExtensionFilter filter = new FileNameExtensionFilter(
                                "txt Files", "txt");
                fileChooser.setFileFilter(filter);
                int index = fileChooser.showDialog(null, "Open File");
                if (index == JFileChooser.APPROVE_OPTION) {
                        File[] files = fileChooser.getSelectedFiles();
                        for (int i = 0; i < files.length; i++) {
                                listModel.addElement(files[i]);
                        }
                }
        }
}
private class selectMutantAction implements ActionListener {
        @SuppressWarnings("unchecked")
        public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setMultiSelectionEnabled(true);
                String startPath = System.getProperty("user.dir");
                fileChooser.setCurrentDirectory(new File(startPath));
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                FileNameExtensionFilter filter = new FileNameExtensionFilter(
                                "bpel Files", "bpel");
                fileChooser.setFileFilter(filter);
                int index = fileChooser.showDialog(null, "Open File");
                if (index == JFileChooser.APPROVE_OPTION) {
                        File[] files = fileChooser.getSelectedFiles();
                        for (int i = 0; i < files.length; i++) {
                                listModel2.addElement(files[i]);
                        }
                }
        }
}
private class DeployAction implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                for (int i=0;i<listModel2.getSize();i++){
                        String fileString =(String) listModel2.getElementAt(i).toString();
                        System.out.println(fileString);
                        try {
                                FileCopy.copyFile(fileString, Mainform.BPELpath.getText());
                                deployThread pThread=new deployThread(bpelFilePath);
                                pThread.start();
                                pThread.join();
                                for (int j = 0; j < listModel.getSize(); j++) {
                                    File file = (File) listModel.get(j);
                                    String testcaseFilePath = file.getAbsolutePath();
                                    String
MutantName=fileString.substring(fileString.lastIndexOf("\\")+1,fileString.lastIndexOf(".")) ;
                                    System.out.println(MutantName);
                                    runThread rThread=new runThread(endpoint, operationName,
testcaseFilePath,MutantName);
                                    rThread.start();
                                }
                        } catch (IOException | InterruptedException e1) {
                                e1.printStackTrace();
                        }
}
        }
}
}
package com.ustb.bpel.view;
import javax.swing.JPanel;
import javax.swing.BorderFactory;
import javax.swing.JFileChooser;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.DefaultComboBoxModel;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Font;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.dom4j.Document;
import org.dom4j.Element;
import alan.testcase.generator.Message;
import alan.testcase.generator.Param;
import alan.testcase.generator.Parameter;
import alan.testcase.generator.TestCaseGenerator;
import alan.testcase.generator.TestCaseInfo;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.File;
import javax.swing.border.LineBorder;
import java.awt.SystemColor;
import java.awt.Color;
import javax.swing.border.TitledBorder;
import javax.swing.UIManager;
public class InputDataGenerator extends JPanel {
        private JTextField filePathTextField;
```

```
    private JButton openButton;
    private JComboBox messageComboBox = null;
    private DefaultComboBoxModel defaultComboBoxModel = null;
    JLabel[] nameLabels = null;
    JTextField[] textFields1 = null;
    JTextField[] textFields2 = null;
    JLabel[] typeLabels = null;
    JComboBox[] signalComboBox1 = null;
    JComboBox[] signalComboBox2 = null;
    private JLabel testCaseNumberLabel = null;
    private JTextField testCaseNumberTextField = null;
    private JTextField savePathTextField = null;
    private JLabel expectedPathLabel = null;
    private JTextField expectedPathTextField = null;
    private JLabel expectedResultLabel = null;
    private JTextField expectedResultTextField = null;
    private JComboBox expectedResultTypeComboBox = null;
    private JButton browseButton = null;
    private JButton generateButton = null;
    private JButton clearButton = null;
    private List<Message> messages = new ArrayList<Message>();
    String[] signal = new String[] { "<", "<=", "!=", "=" };
    String[] signalStr = new String[] { "!=", "=" };
    String[] options = new String[] { "string", "char", "int", "double",
            "float" };
    public InputDataGenerator() {
        setBorder(new LineBorder(SystemColor.controlDkShadow, 2));
        this.setLayout(null);
        initControl();
        initLayout();
    }
    private void initControl() {
        defaultComboBoxModel = new DefaultComboBoxModel();
        messageComboBox = new JComboBox(defaultComboBoxModel);
        messageComboBox.setFont(new Font("Dialog", Font.PLAIN, 12));
        messageComboBox.setBounds(525, 20, 183, 21);
        expectedPathLabel = new JLabel("Expected Path: ");
        expectedPathTextField = new JTextField(18);
        savePathTextField = new JTextField(20);
        savePathTextField.setFont(new Font("Dialog", Font.PLAIN, 12));
        savePathTextField.setBounds(10, 23, 206, 21);
        savePathTextField.setText(getOutputPath());
        browseButton = new JButton("Browse");
        browseButton.setFont(new Font("Dialog", Font.PLAIN, 12));
        browseButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                String startPath = System.getProperty("user.dir");
                fileChooser.setCurrentDirectory(new File(startPath));
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
```

```
                        int index = fileChooser.showDialog(null, "Open File");
                        if (index == JFileChooser.APPROVE_OPTION) {
                              savePathTextField.setText(fileChooser.getSelectedFile()
                                          .getAbsolutePath());
                        }
                  }
            });
            browseButton.setBounds(226, 22, 105, 23);
            clearButton = new JButton("Clear");
            clearButton.setFont(new Font("Dialog", Font.PLAIN, 12));
            clearButton.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        testCaseNumberTextField.setText(null);
                        filePathTextField.setText(null);
                        expectedResultTextField.setText(null);
                  }
            });
            clearButton.setBounds(189, 19, 112, 23);
            openButton = new JButton("Browse\r\n");
            openButton.setFont(new Font("Dialog", Font.PLAIN, 12));
            openButton.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        JFileChooser fileChooser = new JFileChooser();
                        String startPath = System.getProperty("user.dir");
                        fileChooser.setCurrentDirectory(new File(startPath));
                        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                        int index = fileChooser.showDialog(null, "Open File");
                        if (index == JFileChooser.APPROVE_OPTION) {
                              filePathTextField.setText(fileChooser.getSelectedFile()
                                          .getAbsolutePath());
                        }
                        ParseWSDL();
                  }
            });
            openButton.setBounds(343, 20, 141, 23);
            filePathTextField = new JTextField(30);
            filePathTextField.setBounds(166, 20, 150, 21);
            generateButton = new JButton("Generate");
            generateButton.setFont(new Font("Dialog", Font.PLAIN, 12));
            generateButton.setBounds(46, 19, 96, 23);
            messageComboBox.addActionListener(new SelectAction());
            generateButton.addActionListener(new GenerateAction());
      }
      private void initLayout() {
            this.setLayout(null);
            setLayout(null);
            this.add(getOpenPanel());
            this.add(getInputParameterPanel());
            {
            }
```

```
	}
	private JPanel getOpenPanel() {
		JPanel panel = new JPanel();
		panel.setBounds(10, 10, 727, 56);
		panel.setBorder(new TitledBorder(UIManager.getBorder("TitledBorder.border"),
"Specify File", TitledBorder.CENTER, TitledBorder.TOP, null, null));
		panel.setLayout(null);
		panel.add(filePathTextField);
		panel.add(openButton);
		panel.add(messageComboBox);
		{
			JLabel lblNewLabel_1 = new JLabel("BPEL's WSDL");
			lblNewLabel_1.setFont(new Font("Dialog", Font.PLAIN, 12));
			lblNewLabel_1.setBounds(24, 23, 132, 15);
			panel.add(lblNewLabel_1);
		}
		return panel;
	}
	private JPanel getExpectedOutputPanel() {
		JPanel panel = new JPanel();
		panel.setBounds(10, 22, 341, 52);
		panel.setBorder(BorderFactory.createTitledBorder("Expected Output"));
		panel.setLayout(null);
		testCaseNumberTextField = new JTextField(18);
		testCaseNumberTextField.setBounds(165, 24, 48, 18);
		panel.add(testCaseNumberTextField);
		{
			JLabel lblNewLabel = new JLabel("Number of Test Case: ");
			lblNewLabel.setFont(new Font("Dialog", Font.PLAIN, 12));
			lblNewLabel.setBounds(10, 27, 145, 15);
			panel.add(lblNewLabel);
		}
return panel;
	}
	private JPanel getBrowsePanel() {
		JPanel panel = new JPanel();
		panel.setBounds(10, 84, 341, 61);
		panel.setBorder(BorderFactory.createTitledBorder("Save Path"));
		panel.setLayout(null);
		panel.add(savePathTextField);
		panel.add(browseButton);
		return panel;
	}
	private JPanel getRightPanel() {
		JPanel panel = new JPanel();
		panel.setBounds(332, 28, 361, 257);
		panel.setLayout(null);
		panel.add(getBrowsePanel());
		panel.add(getButtonPanel());
		panel.add(getExpectedOutputPanel());
		return panel;}
	private void ParseWSDL() {
		String variable = "";
		String type = "";
		String midPath = "";
		String path = "";
		int index;
		String filePath = filePathTextField.getText();
		Mainform.textArea_1.append("Open the WSDL file:"+"---"+filePath+ "\r\n");
		Document document = alan.converter.core.XMLHelper.openXMLFile(filePath);
		Element root = document.getRootElement();
		for (Iterator iter = root.elementIterator(); iter.hasNext();) {
			Element element = (Element) iter.next();
			if (element.getName().equals("message")) {
				Message m = new Message();
				m.setName(element.attributeValue("name"));
				for (Iterator ii = element.elementIterator(); ii.hasNext();) {
					Element ele = (Element) ii.next();
					Parameter p = new Parameter();
					p.setName(ele.attributeValue("name"));
					p.setType(getTypeFromXMLSchem(ele.attributeValue("type")));
					m.getParameters().add(p);
				}
				defaultComboBoxModel.insertElementAt(m.getName(), 0);
				messages.add(m);
			}
		}
		paint();
	}
	private JPanel getLeftPanel() {
		JPanel panel = new JPanel();
		panel.setBounds(22, 28, 300, 257);
		panel.setPreferredSize(new Dimension(300, 160));
		panel.setBorder(BorderFactory.createTitledBorder("Input Constraint"));
		panel.setLayout(new GridBagLayout());
		GridBagConstraints c = new GridBagConstraints();
		c.gridwidth = 1;
		c.gridheight = 1;
		c.gridx = 0;
		c.gridy = 0;
		if (nameLabels != null && textFields1 != null && textFields2 != null) {
			for (int i = 0; i < nameLabels.length; i++) {
				panel.add(textFields1[i], c);
				c.gridx++;
				panel.add(signalComboBox1[i], c);
				c.gridx++;
				panel.add(nameLabels[i], c);
				c.gridx++;
				panel.add(signalComboBox2[i], c);
				c.gridx++;
```

```
					panel.add(textFields2[i], c);
					c.gridx++;
					panel.add(typeLabels[i], c);
					c.gridx = 0;
					c.gridy++;
				}
			}
			return panel;
		}
		private JPanel getInputParameterPanel() {
			JPanel panel = new JPanel();
			panel.setBorder(new TitledBorder(null, "Output", TitledBorder.CENTER, TitledBorder.TOP, null, null));
			panel.setBounds(10, 76, 727, 295);
			panel.setLayout(null);
			panel.add(getLeftPanel());
			panel.add(getRightPanel());
			return panel;
		}
		private void paint() {
			this.removeAll();
			this.add(getOpenPanel(), BorderLayout.NORTH);
			this.add(getInputParameterPanel(), BorderLayout.SOUTH);
		}
		private String getTypeFromXMLSchem(String type) {
			String ret = "";
			if (type.equals("xsd:string")) {
				ret = "string";
			} else if (type.equals("xsd:int")) {
				ret = "int";
			} else if (type.equals("xsd:double")) {
				ret = "double";
			} else if (type.equals("xsd:float")) {
				ret = "float";
			} else if (type.equals("xsd:char")) {
				ret = "char";
			}
			return ret;
		}
		private class SelectAction implements ActionListener {
			public void actionPerformed(ActionEvent e) {
				String message = (String) messageComboBox.getSelectedItem();
				int index = getIndexOfMessages(message);
				int length = messages.get(index).getParameters().size();
				nameLabels = new JLabel[length];
				textFields1 = new JTextField[length];
				signalComboBox1 = new JComboBox[length];
				signalComboBox2 = new JComboBox[length];
				textFields2 = new JTextField[length];
				typeLabels = new JLabel[length];
```

```
                for (int i = 0; i < length; i++) {
                    Parameter p = (Parameter) messages.get(index).getParameters()
                            .get(i);
                    if (p.getType().equals("string")) {
                        textFields1[i] = new JTextField(3);
                        textFields1[i].setEditable(false);
                        signalComboBox1[i] = new JComboBox(signalStr);
                        signalComboBox1[i].setEnabled(false);
                        nameLabels[i] = new JLabel(p.getName());
                        signalComboBox2[i] = new JComboBox(signalStr);
                        textFields2[i] = new JTextField(3);
                        typeLabels[i] = new JLabel(p.getType());
                    } else {
                        textFields1[i] = new JTextField(3);
                        signalComboBox1[i] = new JComboBox(signal);
                        nameLabels[i] = new JLabel(p.getName());
                        signalComboBox2[i] = new JComboBox(signal);
                        textFields2[i] = new JTextField(3);
                        typeLabels[i] = new JLabel(p.getType());
                    }
                }
                paint();
            }
        }
        private String getOutputPath() {
            return System.getProperty("user.dir") + "\\testcase\\"+ "testcases.txt";
        }
        private int getIndexOfMessages(String message) {
            int ret = -1;
            for (int i = 0; i < messages.size(); i++) {
                if (message.equals(messages.get(i).getName())) {
                    ret = i;break;
                }
            }
            return ret;
        }
        private class GenerateAction implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                TestCaseInfo tci = new TestCaseInfo();
                int number = nameLabels.length;
                for (int i = 0; i < number; i++) {
                    String literal = "";
                    String type = typeLabels[i].getText();
                    if (type.equals("string")) {
                        literal += nameLabels[i].getText();
                        literal += signalComboBox2[i].getSelectedItem().toString();
                        literal += "\"" + textFields2[i].getText() + "\"";
                    } else {
                        literal += textFields1[i].getText();
                        literal += signalComboBox1[i].getSelectedItem().toString();
                        literal += nameLabels[i].getText();
                        literal += signalComboBox2[i].getSelectedItem().toString();
                            literal += textFields2[i].getText();
                    }
                    Param p = new Param();
                    p.setLiteral(literal);
                    p.setType(type);
                    Mainform.textArea_1.append("The parameter:"+"---"+p.toString()+ "\r\n");
                    tci.getParams().add(p);
                }
                Param p = new Param();
                p.setLiteral("null");
                p.setType("string");
                tci.setExpectedResult(p);
                tci.setExpectedPath(expectedPathTextField.getText());
                tci.setTestcaseNumber(Integer.parseInt(testCaseNumberTextField
                        .getText()));
                TestCaseGenerator.getInstance().generateRandomTestCase(tci,
                        savePathTextField.getText());
            }
        }
}
package alan.testcase.executor;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
        public class FileCopy {
            public   static   void   copyFile(String   file1,String   file2)throws   IOException{

                    FileInputStream fis=new FileInputStream(file1);
                    FileOutputStream fos=new FileOutputStream(file2);
                    int temp;
                    while((temp=fis.read())!=-1){
                     fos.write(temp);
                    }
                    fis.close();
                    fos.close();
                    System.out.println("从"+file1+"到"+file2);
}
package alan.testcase.executor;
import java.io.IOException;
public class deployThread extends Thread{
    String bpelFilePath;
    public deployThread(String bpelFilePath){
        this.bpelFilePath=bpelFilePath;
    }
    public void run(){
Runtime deployrt = Runtime.getRuntime();
                Process deployProcess;
```

```
                    try {
        deployProcess = deployrt.exec("cmd /c start ant -f " + bpelFilePath +"\n"+ "deploy");
                        try {sleep(60000);
                        } catch (InterruptedException e)
                  {
                            e.printStackTrace();
                        }
                        try {
        deployProcess.waitFor();
                        } catch (InterruptedException e1) {
                            e1.printStackTrace();
                        }
                          int i = deployProcess.exitValue();
                          if (i == 0) {
                            System.out.println("执行完成.") ;
                          } else {
                            System.out.println("执行失败.") ;
                          }
                          deployProcess.destroy();
                          deployProcess = null;
                    } catch (IOException e2) {
                        e2.printStackTrace();
                    }
                      killProcess();
    }
              public void killProcess(){
                Runtime rt = Runtime.getRuntime();
                Process p = null;
                try {
                  rt.exec("cmd.exe /C start wmic process where name='cmd.exe' call terminate");
                } catch (IOException e)
                  {
                  e.printStackTrace();
                }
              }
}
package alan.testcase.executor;
import java.util.List;
public class runThread extends Thread{
    String   operationName;
    String endpoint;
    String testcaseFilePath;
    String   MutantName;
    public runThread(String endpoint, String operationName,
              String testcaseFilePath, String MutantName)
{
        this.endpoint=endpoint;
        this.operationName=operationName;
        this.testcaseFilePath=testcaseFilePath;
        this.MutantName=MutantName;
```

```
	}
	public void run()
{
		runBPELTestClient(endpoint, operationName, testcaseFilePath,MutantName);
		try {
		sleep(3000);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}
	private void runBPELTestClient(String endpoint, String operationName,
			String testcaseFilePath,String MutantName)
{
			RunCase executor = new RunCase();
			List<ExecutionResult> executionResults = executor.execute(endpoint,
					operationName, testcaseFilePath,MutantName);
			Object[] rowData = new Object[7];
			for (int j = 0; j < executionResults.size(); j++) {
				ExecutionResult er = executionResults.get(j);
				rowData[0] = er.getNo();
				rowData[1] = er.getBeginTime();
				rowData[2] = er.getEndTime();
				rowData[3] = er.getInput();
				rowData[4] = er.getExpectedResult();
				rowData[5] = er.getExecutedResult();
				rowData[6] = er.getStatus();
			}
	}
}
package operator;
import java.util.Arrays;
import java.util.List;import org.dom4j.Element;
import parser.BPELMutator;
import parser.mu_List;
public class EAA extends BPELMutator {
	List<String> arithList = Arrays.asList("+", "-", "*", "div", "mod");
	public EAA() {
		super();
	}
	public void visit(Element element) {
		if (element.getName().equals("condition")
				|| element.getName().equals("transitionCondition")
				|| element.getName().equals("joinCondition")
				|| element.getName().equals("until")
				|| element.getName().equals("for")
				|| element.getName().equals("repeatEvery")
				|| element.getName().equals("startCounterValue")
				|| element.getName().equals("finalCounterValue")
				|| element.getName().equals("branches")
				|| element.getName().equals("from")
```

```
                    || element.getName().equals("to")) {
                    String expression = element.getText();
                int index = -1;
                if (expression.contains("+"))
                    index = 0;
                else if (expression.contains("-"))
                    index = 1;
                else if (expression.contains("*"))
                    index = 2;
                else if (expression.contains("/"))
                    index = 3;
                else if (expression.contains("%"))
                    index = 4;
                if (index >= 0) {
                    for (int i = 0; i < arithList.size(); i++) {
                        if (i != index) {
                            muID++;
                            mu_List mu_List = new mu_List("EAA", element,
                                    new EAA_Writer(element, arithList.get(index),
                                            arithList.get(i)), muID);
                            MutantsList.add(mu_List);
                        }
}
}
            } else
                super.visit(element);
        }
}
package operator;
import org.dom4j.Element;
import parser.BPELMutator;
import parser.mu_List;
public class XMC extends BPELMutator {
    public XMC()
{
        super();
}
    public void visit(Element element) {
        if (element.getName().equals("compensationHandler")) {
            System.out.println("find " + element.getName());
            muID++;
            mu_List mu_List = new mu_List("XMC", element, new XMC_Writer(
                    element), muID);
            MutantsList.add(mu_List);
        }
    else{
            super.visit(element);
    }
}
}
```