Security Item Risk Valuation and Assessment Tool

A Security Case Study submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Prepared By
Derek J. Sedlack
Division of Information Sciences
Graduate School of Computer and Information Sciences
Nova Southeastern University

Conducted for
DISS 799 Information Security
Dr. James Cannady, Professor

2005

Certification of Authorship of Doctoral Work

Submitted to:
Professor James Cannady, Ph.D.

Student's Name:
Derek J. Sedlack

Date of Submission:
September 25, 2005

Purpose and Title of Submission:
Partial fulfillment requirement for doctoral degree.
Assignment 2: Security Risk

Certification of Authorship: I hereby certify that I am the author of this document and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas, or words that are copied directly or paraphrased in the document. Sources are properly credited according to accepted standards for professional publications. I also certify that this paper was prepared by me for this purpose.

Student's Signature:_____

Security Item Risk Valuation and Assessment Tool

1. Identify the problem that you were trying to solve.

While Patterson (2005) discussions surround developing a global corporate security policy and enforcing the top-down aspect of security aspects, Return on Security Investment (ROSI) is part of the foundation of any good security implementation. Companies are just beginning to devise critical security plans, but fail to account for asset security as a cost-intensive labor similar to the traditional accounting practices associated with depreciating resources. Peltier (2001) notes a plethora of security tables, but establishes no rubric for corporations to apply, or use to begin planning a more applicable security plan that includes the actual costs of security assets.

2. Justify the need for an information security-based approach to solve the problem.

Creating a database of GAAP cost analysis based on security will help companies assess current expenses, create comparison lists to reduce expenditures, and potentially provide a method of comparative security risk analysis that can span divisions and intercontinental offices. Building such a list will provide executive management with a more definitive expense, risk, and assessment report that affords better decision making, strategic planning, and disaster recovery techniques. Dr. Cannady noted during lecture the idea of spending $50,000 to secure a $1,000 item is commonplace and an irresponsible use of capital unless the item is vital to corporate survival, thus requiring a method of distinction to include this paradigm.

3. Discuss the problem solving methodology in detail.

In following the many lists provided by Peltier (2001) and preliminary research, I believed that phase 1 should cover simple GAAP security risk reporting and item risk assessments based on weighted environmental conditions. Advanced features that would be vital to properly developing a broad security risk assessment is included in the attached spreadsheet and scheduled to be included during phase 2.

Risk assessments should cover a broad integer range that can produce a corporate risk table, or bar graph, that will show the most expensive items to cover versus those that require the least expenditures. While manual comparisons are included in phase 1, a programmatic approach is planned for future implementations.

4. Provide the design specifications for the system

Risk valuations, as defined by Peltier (2001), includes the following core components:

1. Name of the item
2. Value of the item (this can be further segmented into replacement/original cost)
3. Risk factors gauged from Low to High on a Likert scale based on "*best-guess*":
   - Priority – importance of the item to business continuity
   - Loss impact – difficulty for item replacement
   - Loss expectancy – how often insurance is required
4. Safeguard Costs are an aggregate of the actual costs to the following:
   - Avoidance costs
   - Assurance costs
   - Detection costs
   - Recovery costs
5. Potential Annual Premium – this is a potential cost of asset insurance based on

*((value * risk factors) + safeguard costs)*

Risk assessments are more complicated, but phase 1 calculations are based on the following components:

1. Name of the item (future implementations carry over the items name from the Risk Valuation Tab)
2. Multipliers – method of comparing higher likelihoods of incident
3. Threat Table:
   - Natural disaster – aggregate of each potential occurrence
   - Accidental disaster – aggregate of each potential occurrence
   - Deliberate disaster – aggregate of each potential occurrence
4. Single Item Risk Assessment – this is a comparative based on selected conditions of risk based on

*(natural * multiplier) + (accidental * multiplier) + (deliberate * multiplier)*

References

Patterson, T. (2005). *Mapping Security: The Corporate Security Sourcebook for Today's Global Economy*. Addison Wesley Professional, Boston, Massachusetts.

Peltier, T. R. (2001). *Information Security Risk Analysis*. CRC Press LLC, Boca Raton, Florida.

Appendix A: Source Code

```
/*************************************************************************
        ****************/
/*                                                                       */
/* Created by Derek J. Sedlack                                           */
/* Address: 5043 Solar Point Drive                                       */
/*          Greenacres, Florida 33463                                    */
/* Email: Sedlack@Nova.edu                                               */
/* Web: scis.nova.edu/~sedlack                                           */
/*                                                                       */
/*************************************************************************
        ****************/
/*                                                                       */
/* DISS 799 Information Security Management                              */
/* Dr. James Cannady                                                     */
/* Graduate School of Computer and Information Sciences                  */
/* Nova Southeastern University                                          */
/*                                                                       */
/*************************************************************************
        ****************/
/*                                                                       */
/* This program should help establish a simple GAAP estimation of how much   */
/* costs should be associated with the insurance of an item that is critical  */
/* to a business. It also provides a method of comparing multiple items by    */
/* assigning disaster analysis, but only currently for single items (both).   */
/*                                                                       */
/* Future consideration should include databasing items for charting comparisons  */
/* and multiple valuations of corporate assets.                          */
/*                                                                       */
/*************************************************************************
        ****************/
/*                                                                       */
/* This program is copyrighted and protected by U.S. Law and may not be used, copied,
        */
/* or distributed without the expressed permission of the author, partially or in total.*/
/*                                                                       */
/*************************************************************************
        ****************/


import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import javax.swing.BorderFactory.*;
```

```
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeEvent;
import java.text.*;
import java.lang.Integer;

public class RiskAnalysis extends JPanel implements PropertyChangeListener,
        ActionListener
{
 // IRV = Item Risk Valuation
 // IRV Labels to identify the fields and buttons
 private JLabel  IRV_ItemNameLabel;
 // private JLabel  IRA_ItemNameLabel;
 private JLabel  IRV_ValueLabel;
 private JLabel  IRV_RiskFactorLabel;
 private JLabel  IRV_SafeguardCostLabel;
 private JLabel  IRV_PotentialAnnualPremiumLabel;
 private JLabel  IRV_PriorityLabel;
 private JLabel  IRV_LossImpactLabel;
 private JLabel  IRV_LossExpectancyLabel;
 private JLabel  IRV_InfoLabel;
 private JLabel  IRV_PotentialAnnualPremiumCalcLabel;
 private JLabel  IRV_RiskFactorCalcLabel;

 // IRA = Item Risk Assessment
 // IRA Labels to identify the fields and buttons
 private JLabel  IRA_ND_MultiplierLabel;
 private JLabel  IRA_A_MultiplierLabel;
 private JLabel  IRA_D_MultiplierLabel;
 private JLabel  IRA_NaturalDisasterValuationLabel;
 private JLabel  IRA_AccidentValuationLabel;
 private JLabel  IRA_DeliberateValuationLabel;
 private JLabel  IRA_ND_ValuationCalcLabel;
 private JLabel  IRA_A_ValuationCalcLabel;
 private JLabel  IRA_D_ValuationCalcLabel;
 private JLabel  IRA_SingleItemRiskValuationCalcLabel;
 private JLabel  IRA_InfoLabel;



 // strings for Item Risk Valuation Labels
 private static String IRV_ITEM_NAME          = "Item Name : ";
 private static String IRV_VALUE              = "Value: ";
```

```
private static String IRV_RISK_FACTOR            = "Risk Factor: ";
private static String IRV_SAFEGUARD_COST         = "Safeguard Cost: ";
private static String IRV_PRIORITY               = "Priority: ";
private static String IRV_LOSS_IMPACT            = "Loss Impact: ";
private static String IRV_LOSS_EXPECTANCY        = "Loss Expectancy: ";
private static String IRV_CALCULATE              = "Calculate";
private static String IRV_CLEAR                  = "Clear";
private static String IRV_POTENTIAL_ANNUAL_PREMIUM = "Potential Annual
        Premium ";


// strings for Item Risk Assessment Labels

private static String IRA_MULTIPLIER             = "Multiplier";
private static String IRA_NATURAL_DISASTER       = "Natural Disaster";
private static String IRA_ACCIDENTAL             = "Accidental";
private static String IRA_DELIBERATE             = "Deliberate";
private static String IRA_CALCULATE              = "Calculate";
private static String IRA_CLEAR                  = "Clear";
private static String IRA_SINGLE_ITEM_RISK_VALUATION  = "Single Item Risk
        Valuation";
private static String IRA_ND_RISK_VALUATION      = "Natural Disaster Risk
        Valuation";
private static String IRA_A_RISK_VALUATION       = "Accident Risk Valuation";
private static String IRA_D_RISK_VALUATION       = "Deliberate Risk Valuation";



// Arrays used for Item Risk Valuation
private static String[] HighLowRanges = {"Low", "Low Medium", "Medium", "Medium
        High", "High"};

// There is 1:1 correspondance between the index values for the DateRanges displayed
// in the combobox (DateRanges), and the values in DateValueRanges used to
// calculate risk factor
private static String[] DateRanges = {"Daily", "Weekly", "Monthly", "Twice / Year",
                              "Yearly", "2 Years", "5 Years", "10 Years",
                              "50 years","100 Years", "200 Years",
                              "300 Years", "Never" };
private static double[] DateValueRanges = {365.33, 52, 12, 2, 1, 0.5, 0.2, 0.04, 0.02,
        0.01, 0.005, 0.00333, 0 };



// Arrays used for Item Risk Assessment
```

```
private static String[] NaturalDisasterList = {"Earthquake", "Flooding", "Hurricane",
        "Landslide", "Sandstorm",
                                "Snow/Ice Storm", "Tornado", "Tsunami", "Volcano",
        "Windstorm" };
private static String[] AccidentalList = {"Disclosure", "Electrical Disturbance",
        "Electrical Interruption",
                                "Emanation", "Fire", "Hardware Failure", "Liquid Leakage",
                                "Software Error", "Telecom Interruption" };

private static String[] DeliberateList = {"Alteration of Data", "Alteration of Software",
                                "Bomb Threat", "Disclosure", "Sabotage", "Fraud",
                                "Riot/Civil Disorder", "Strike", "Theft",
                                "Vandalism", "Terrorism" };


// Index values for Item Risk Valuation
private static int HI_LOW_START_INDEX = 4; // Initiate High/Low combo with High
private static int DATE_START_INDEX   = 4; // Initiate Date combo with Yearly, isn't
        this the most common?


// Entry fields and Combo boxes for Item Risk Valuation
private JFormattedTextField IRV_ItemNameEntryField;
private JFormattedTextField IRV_ValueEntryField;
private JFormattedTextField IRV_SafeguardCostEntryField;
private JComboBox         IRV_LossExpectancyCombo;
private JComboBox         IRV_PriorityCombo;
private JComboBox         IRV_LossImpactCombo;

// Entry fields and Ceck boxes for Item Risk Assessment
private JFormattedTextField IRA_ND_MultiplierEntryField; // multiplier for natural
        disaster
private JFormattedTextField IRA_A_MultiplierEntryField;  // multiplier for accidental
private JFormattedTextField IRA_D_MultiplierEntryField;  // multiplier for disaster

private JCheckBox IRA_ND_EarthquakeCheckBox;
private JCheckBox IRA_ND_FloodingCheckBox;
private JCheckBox IRA_ND_HurricaneCheckBox;
private JCheckBox IRA_ND_LandslideCheckBox;
private JCheckBox IRA_ND_SandstormCheckBox;
private JCheckBox IRA_ND_SnowIceStormCheckBox;
private JCheckBox IRA_ND_TornadoCheckBox;
private JCheckBox IRA_ND_TsunamiCheckBox;
private JCheckBox IRA_ND_VolcanoCheckBox;
private JCheckBox IRA_ND_WindstormCheckBox;
```

```
private JCheckBox[] NaturalDisasterCheckBoxList = {IRA_ND_EarthquakeCheckBox,
                                    IRA_ND_FloodingCheckBox,
                                    IRA_ND_HurricaneCheckBox,
                                    IRA_ND_LandslideCheckBox,
                                    IRA_ND_SandstormCheckBox,
                                    IRA_ND_SnowIceStormCheckBox,
                                    IRA_ND_TornadoCheckBox,
                                    IRA_ND_TsunamiCheckBox,
                                    IRA_ND_VolcanoCheckBox,
                                    IRA_ND_WindstormCheckBox};

private JCheckBox IRA_A_DisclosureCheckBox;
private JCheckBox IRA_A_ElectricalDisturbanceCheckBox;
private JCheckBox IRA_A_ElectricalInterruptionCheckBox;
private JCheckBox IRA_A_EmanationCheckBox;
private JCheckBox IRA_A_FireCheckBox;
private JCheckBox IRA_A_HardwareFailureCheckBox;
private JCheckBox IRA_A_LiquidLeakageCheckBox;
private JCheckBox IRA_A_SoftwareErrorCheckBox;
private JCheckBox IRA_A_TelecomInterruptionCheckBox;

private JCheckBox[] AccidentalCheckBoxList = {IRA_A_DisclosureCheckBox,
                                IRA_A_ElectricalDisturbanceCheckBox,
                                IRA_A_ElectricalInterruptionCheckBox,
                                IRA_A_EmanationCheckBox,
                                IRA_A_FireCheckBox,
                                IRA_A_HardwareFailureCheckBox,
                                IRA_A_LiquidLeakageCheckBox,
                                IRA_A_SoftwareErrorCheckBox,
                                IRA_A_TelecomInterruptionCheckBox};


private JCheckBox IRA_D_AlterationOfDataCheckBox;
private JCheckBox IRA_D_AlteraionOfSoftwareCheckBox;
private JCheckBox IRA_D_BombThreatCheckBox;
private JCheckBox IRA_D_DisclosureCheckBox;
private JCheckBox IRA_D_SabotageCheckBox;
private JCheckBox IRA_D_FraudCheckBox;
private JCheckBox IRA_D_RiotCivilDisorderCheckBox;
private JCheckBox IRA_D_StrikeCheckBox;
private JCheckBox IRA_D_TheftCheckBox;
private JCheckBox IRA_D_VandalismCheckBox;
private JCheckBox IRA_D_TerrorismCheckBox;
```

```
private JCheckBox[] DeliberateCheckBoxList = { IRA_D_AlterationOfDataCheckBox,
                                               IRA_D_AlteraionOfSoftwareCheckBox,
                                               IRA_D_BombThreatCheckBox,
                                               IRA_D_DisclosureCheckBox,
                                               IRA_D_SabotageCheckBox,
                                               IRA_D_FraudCheckBox,
                                               IRA_D_RiotCivilDisorderCheckBox,
                                               IRA_D_StrikeCheckBox,
                                               IRA_D_TheftCheckBox,
                                               IRA_D_VandalismCheckBox,
                                               IRA_D_TerrorismCheckBox };

// Buttons used for Item Risk Valuation
private JButton IRV_CalculateButton;
private JButton IRV_ClearButton;

// Buttons used for Item Risk Assessment
private JButton IRA_CalculateButton;
private JButton IRA_ClearButton;


// Item Risk Valuation (IRV) variables that contains customer's input
private String IRV_ItemNameInput       = "";
private double IRV_ValueInput          = 0;
private double IRV_SafeguardCostInput  = 0;
private double IRV_PotentialAnnualPremiumCalcValue = 0;
private double IRV_RiskFactorCalcValue = 0;
private int    IRV_PriorityInput       = HI_LOW_START_INDEX+1; // (High = 5,
        MedHigh = 4, Med = 3, MedLow = 2, Low = 1),
private int    IRV_LossImpactInput     = HI_LOW_START_INDEX+1; // (High = 5,
        MedHigh = 4, Med = 3, MedLow = 2, Low = 1),
private double IRV_LossExpectancyInput =
        DateValueRanges[DATE_START_INDEX];  // (10 Years = 0.04, 100 years...
private int    IRV_Count               = 0;


private JTabbedPane tabbedPane;
private boolean debug = false;


public RiskAnalysis()
{

  super(new BorderLayout());
```

```
 JComponent panel1 = makeInformationPanel();
 tabbedPane = new JTabbedPane(JTabbedPane.TOP,
        JTabbedPane.SCROLL_TAB_LAYOUT);


 // Information Panel

 tabbedPane.addTab("Information", null, panel1, "Neither does this...not a
        programmer");
 tabbedPane.setToolTipTextAt(0, "This doesn't work");

 //ALT + I to jump to Information Page
 tabbedPane.setMnemonicAt(0, KeyEvent.VK_I);


 JComponent panel2 = makeItemRiskValuationPanel();
 tabbedPane.addTab("Item Risk Valuation", panel2);
 //ALT + V to jump to Valuation Page
 tabbedPane.setMnemonicAt(1, KeyEvent.VK_V);
 tabbedPane.setToolTipTextAt(1, "Again..need some help here");

 JComponent panel3 = makeItemRiskAssessmentPanel();
 tabbedPane.addTab("Item Risk Assessment", panel3);
 //ALT + A to jump to Assessment Page
 tabbedPane.setMnemonicAt(2, KeyEvent.VK_A);

 tabbedPane.setPreferredSize(new Dimension(900, 500));

 // Add the tabbed pane to this panel.
 add(tabbedPane);


}

 public void propertyChange(PropertyChangeEvent e) {

     Object source = e.getSource();

     if (debug) {
       System.out.println ("Enter PropertyChange() for " + source.toString() + "\n");
     }

     if (IRV_Count < 3 ) {
       IRV_Count++;
       return;
```

```
}

if (source == IRV_ItemNameEntryField) {

  IRV_ItemNameInput = IRV_ItemNameEntryField.getValue().toString();
  // IRA_ItemNameLabel.setText(IRV_ItemNameInput);
}
else if ( source == IRV_ValueEntryField )
{
  try
  {
    String irv_value = IRV_ValueEntryField.getValue().toString();
    if (irv_value == null || !irv_value.equals("")) {
      IRV_ValueInput = Double.parseDouble(irv_value);
      IRV_ValueLabel.setForeground (Color.black);
      IRV_InfoLabel.setText ("");
    }
  }
  catch (NumberFormatException nfe)
  {

    IRV_InfoLabel.setText ("Must be numeric.");
    IRV_InfoLabel.setForeground (Color.red);
    IRV_ValueLabel.setForeground (Color.red);
    IRV_ValueEntryField.setValue ("");

  }

} else if (source == IRV_SafeguardCostEntryField) {
  try
  {
    String irv_safeguard = IRV_SafeguardCostEntryField.getValue().toString();
    if (irv_safeguard == null || !irv_safeguard.equals("")) {
      IRV_SafeguardCostInput =
   Double.parseDouble(IRV_SafeguardCostEntryField.getValue().toString());
      IRV_SafeguardCostLabel.setForeground (Color.black);
      IRV_InfoLabel.setText ("");
    }
  }
  catch (NumberFormatException nfe)
  {

    IRV_InfoLabel.setText ("Must be numeric.");
    IRV_InfoLabel.setForeground (Color.red);
    IRV_SafeguardCostLabel.setForeground (Color.red);
```

```
            IRV_SafeguardCostEntryField.setValue("");

          }
        }

    }


public void actionPerformed(ActionEvent e) {
      if (debug) {
        System.out.println ("enter actionPerformed.  ActionCommand = " +
         e.getActionCommand());
        System.out.println ("e.getSource() = " + e.getSource());
        System.out.println ("tabbedPane = " + tabbedPane.getSelectedIndex());
      }

      if (tabbedPane.getSelectedIndex() == 1) {
        actionPerformedOnItemRiskValuationPanel(e);
      }
      else if ( tabbedPane.getSelectedIndex() == 2) {
        actionPerformedOnItemRiskAssessmentPanel(e);
      }

    }

private void actionPerformedOnItemRiskValuationPanel(ActionEvent e)
{
   if (IRV_CLEAR.equals(e.getActionCommand())) {
      if (debug) {
        System.out.println ("IRV Clear button selected");
      }

      // clear out values displayed in Entry Fields & ComboBox
      IRV_ItemNameEntryField.setValue("");
      IRV_ItemNameLabel.setForeground (Color.black);
      //IRA_ItemNameLabel.setText("");
      IRV_ValueEntryField.setValue("");
      IRV_ValueLabel.setForeground (Color.black);
      IRV_RiskFactorCalcLabel.setText("");
      IRV_SafeguardCostEntryField.setValue("");
      IRV_SafeguardCostLabel.setForeground (Color.black);
      IRV_PotentialAnnualPremiumCalcLabel.setText("");
      IRV_InfoLabel.setText ("");
      // reset comboBox to default specified for constructor
      IRV_PriorityCombo.setSelectedIndex(HI_LOW_START_INDEX);
```

```
  IRV_LossImpactCombo.setSelectedIndex(HI_LOW_START_INDEX);
  IRV_LossExpectancyCombo.setSelectedIndex(DATE_START_INDEX);

  // reset Input values stored in variables
  IRV_ItemNameInput       = "";
  // IRA_ItemNameLabel.setText("");
  IRV_ValueInput          = 0;
  IRV_SafeguardCostInput  = 0;
  IRV_PriorityInput       = HI_LOW_START_INDEX+1;
  IRV_LossImpactInput     = HI_LOW_START_INDEX+1;
  IRV_LossExpectancyInput = DATE_START_INDEX;
  IRV_RiskFactorCalcValue             = 0;
  IRV_PotentialAnnualPremiumCalcValue = 0;

} else if (IRV_CALCULATE.equals(e.getActionCommand())) {
  if (debug) {
    System.out.println ("calculate button selected");
  }

  String strValue = (String)IRV_ValueEntryField.getValue();

  String strSafeguard = (String)IRV_SafeguardCostEntryField.getValue();

  if (strValue.equals("") ) {

    IRV_InfoLabel.setText ("Required Field(s) missing.");
    IRV_InfoLabel.setForeground (Color.red);
    IRV_ValueLabel.setForeground (Color.red);
  }
  if (strSafeguard.equals("") ) {

    IRV_InfoLabel.setText ("Required Field(s) missing.");
    IRV_InfoLabel.setForeground (Color.red);
    IRV_SafeguardCostLabel.setForeground (Color.red);
  }
  if (debug) {
    System.out.println ("value = " + strValue);
    System.out.println ("LossExpectancy = " + IRV_LossExpectancyInput);
  }

   // Risk Factor = Priority + Impact * Expectancy
  IRV_RiskFactorCalcValue = ( IRV_PriorityInput + IRV_LossImpactInput ) *
     IRV_LossExpectancyInput;

  if (debug) {
```

```
      System.out.println ( "Risk Factor = (Priority + Impact) * Expectancy");
      System.out.println ( IRV_RiskFactorCalcValue + " = (" + IRV_PriorityInput + " +
       " + IRV_LossImpactInput + ") * " + IRV_LossExpectancyInput);
    }

    // Potential Annual  Premium = (Value * Risk) + Safeguard
    IRV_PotentialAnnualPremiumCalcValue = (IRV_ValueInput *
       IRV_RiskFactorCalcValue) + IRV_SafeguardCostInput;
    if (debug) {
      System.out.println ( "Potential Annual Premium = (Value* Risk) + Safeguard");
      System.out.println ( IRV_PotentialAnnualPremiumCalcValue + " = (" +
       IRV_ValueInput + " * " + IRV_RiskFactorCalcValue + ") + " +
       IRV_SafeguardCostInput);
    }


    // populate the workbook area

       IRV_PotentialAnnualPremiumCalcLabel.setText(Double.toString(IRV_Potential
       AnnualPremiumCalcValue));
    IRV_RiskFactorCalcLabel.setText(Double.toString(IRV_RiskFactorCalcValue));
  }
  else if (e.getSource() == IRV_LossImpactCombo) {

    // index+1 corresponds to Low, Low Middle, Middle, etc...
    JComboBox jcb = (JComboBox)e.getSource();
    IRV_LossImpactInput = jcb.getSelectedIndex()+1;

    if (debug) {
      System.out.println ("Loss Impact combo box selection = " +
       jcb.getSelectedItem());
    }

  }
  else if (e.getSource() == IRV_PriorityCombo) {

    // index+1 corresponds to Low, Low Middle, Middle, etc...
    JComboBox jcb = (JComboBox)e.getSource();
    IRV_PriorityInput = jcb.getSelectedIndex()+1;

    if (debug) {
      System.out.println ("Priority combo box selection = " + jcb.getSelectedItem());
    }

  }
```

```
  else if (e.getSource() == IRV_LossExpectancyCombo) {

    JComboBox jcb = (JComboBox)e.getSource();
    int index = jcb.getSelectedIndex();
    IRV_LossExpectancyInput = DateValueRanges[index];

    if (debug) {
      System.out.println ("IRV_LossExpectancyInput = " + IRV_LossExpectancyInput);
    }

  }
}

private void actionPerformedOnItemRiskAssessmentPanel(ActionEvent e)
{
  if (IRA_CLEAR.equals(e.getActionCommand())) {
    if (debug) {
      System.out.println ("IRA Clear button selected");
    }


    // clear Natural Disasters checkboxes and entry field
    for (int i = 0; i <NaturalDisasterCheckBoxList.length; i++) {
      NaturalDisasterCheckBoxList[i].setSelected(false);
    }
    IRA_ND_MultiplierEntryField.setValue("");
    IRA_ND_MultiplierLabel.setForeground(Color.black);

    // clear Accidental checkboxes and entry field
    for (int i = 0; i <AccidentalCheckBoxList.length; i++) {
      AccidentalCheckBoxList[i].setSelected(false);
    }
    IRA_A_MultiplierEntryField.setValue("");
    IRA_A_MultiplierLabel.setForeground(Color.black);

    // clear Deliberate checkboxes and entry field
    IRA_D_MultiplierEntryField.setValue("");
    for (int i = 0; i <DeliberateCheckBoxList.length; i++) {
      DeliberateCheckBoxList[i].setSelected(false);
    }

    IRA_D_MultiplierEntryField.setValue("");
    IRA_D_MultiplierLabel.setForeground(Color.black);
```

```
  // clear out workbook area
  IRA_ND_ValuationCalcLabel.setText("");
  IRA_A_ValuationCalcLabel.setText("");
  IRA_D_ValuationCalcLabel.setText("");
  IRA_SingleItemRiskValuationCalcLabel.setText("");
  IRA_InfoLabel.setText("");


} else if (IRV_CALCULATE.equals(e.getActionCommand())) {
  if (debug) {
    System.out.println ("calculate button selected");
  }

  populateValuationWorkArea( IRA_ND_MultiplierEntryField,
     IRA_ND_MultiplierLabel,
                  IRA_ND_ValuationCalcLabel, NaturalDisasterCheckBoxList);


  populateValuationWorkArea( IRA_A_MultiplierEntryField,
     IRA_A_MultiplierLabel,
                  IRA_A_ValuationCalcLabel, AccidentalCheckBoxList);


  populateValuationWorkArea( IRA_D_MultiplierEntryField,
     IRA_D_MultiplierLabel,
                  IRA_D_ValuationCalcLabel, DeliberateCheckBoxList);

  // Single Item Risk valuation = Natural Disaster Valuation + Accidental Valuation +
     Deliberate Valuation
  // incase one or more of the valuations are not set, catch exception per valuation
  double dnd = 0;
  double da  = 0;
  double dd  = 0;

  dnd = validateData ( IRA_InfoLabel, IRA_ND_MultiplierEntryField,
     IRA_ND_MultiplierLabel, IRA_ND_ValuationCalcLabel);
  da  = validateData ( IRA_InfoLabel, IRA_A_MultiplierEntryField,
     IRA_A_MultiplierLabel, IRA_A_ValuationCalcLabel);
  dd  = validateData ( IRA_InfoLabel, IRA_D_MultiplierEntryField,
     IRA_D_MultiplierLabel, IRA_D_ValuationCalcLabel);


  Double sirv = new Double ( dnd + da + dd);
  // if customer presses calculate without setting multiplier/checkboxes
  if (dnd != 0 || da != 0 || dd != 0 ) {
```

```
      IRA_SingleItemRiskValuationCalcLabel.setText ( sirv.toString() );
      IRA_InfoLabel.setText("");
    }
    else
    {
      IRA_SingleItemRiskValuationCalcLabel.setText ("");
    }

    if (debug) {
      System.out.println ("sirv = " + sirv.toString());
    }
    sirv = null;


  }
}

//
private void populateValuationWorkArea ( JFormattedTextField entryField, JLabel
        entryFieldLabel, JLabel calcLabel, JCheckBox[] jcb )
{
    try
    {
      int    checkBoxSelectedTotal = 0;
      double multiplier            = 0;
      Double sum;

      for (int i = 0; i < jcb.length; i++) {
        if ( jcb[i].isSelected()) {
          checkBoxSelectedTotal++;
        }
      }

      // multiply the number of check boxes selected by the multiplier value input
      String strMultiplier = new String ((String)entryField.getValue());

      if (strMultiplier != null || !strMultiplier.equals("")) {

        multiplier = Double.parseDouble (strMultiplier);

        if (multiplier <= 0 ) {
          entryFieldLabel.setForeground(Color.red);
          entryField.setValue("");
          calcLabel.setText("");
          IRA_InfoLabel.setText ("Must be a number greater than 0.");
```

```
            IRA_InfoLabel.setForeground (Color.red);
          }
          sum = new Double ((double)checkBoxSelectedTotal * multiplier);
          calcLabel.setText(sum.toString());
          entryFieldLabel.setForeground(Color.black);
        }

      }
      catch (NumberFormatException nfe)
      {
        entryFieldLabel.setForeground(Color.red);
        entryField.setValue("");
        calcLabel.setText("");
      }
}

double validateData (JLabel infoLabel, JFormattedTextField entryField, JLabel
        multiLabel, JLabel calcLabel )
{
  double dnd = 0;
  try
  {
    dnd = Double.parseDouble(calcLabel.getText());
  }
  catch (NumberFormatException nfe)
  {
    infoLabel.setText ("Must be a number greater than 0.");
    infoLabel.setForeground (Color.red);
    multiLabel.setForeground(Color.red);
    entryField.setValue("");
    calcLabel.setText("");

  }
  return dnd;
}

private JComponent makeInformationPanel()
{
 JPanel panel = new JPanel(false);

 panel.setLayout(new BorderLayout());
 panel.setBorder(BorderFactory.createTitledBorder("Risk Analysis Calculator
        Overview"));

 JLabel info = new JLabel();
```

```
  String infoText = "<html><br>" +
                "<body bgcolor=e0e0e0>" +
                "Purpose:<br>" +
                "    The devise of this calculator is to better determine key
        security risk indicators:<br>" +
                "       The potential cost of insuring an
        item<br>" +
                "       The potential impact of item loss
        through various threats<br>" +
                "       The relative impact of such loss
        compared to other comparable items<br>" +
                "<br>Item Risk Valuation:<br>" +
                "    Single item valuation depending on item cost,
        multiplied by the risk factor,<br>" +
                "    and added to the cost of safeguarding the item.<br>" +
                "    Risk factor is based on item priority, loss impact, and
        loss expentancy.<br>" +
                "    This should have a database backend to compare
        multiple item valuations.<br>" +
                "<br>" +
                "Item Risk Assessment:<br>" +
                "    Itentify viable threats on the table, assign a multiple (if
        one is more<br>" +
                "    dangerous than another to your organization), then
        select items of<br>" +
                "    valuation to determine comparable levels of
        organizational risk.<br>" +
                "</body>";


   info.setText(infoText);
   panel.add(info, BorderLayout.NORTH);
   return panel;
  }


 private JComponent makeItemRiskValuationPanel()
 {
   JPanel panel = new JPanel(new BorderLayout());
   panel.setLayout(new BorderLayout());
   // create border around page with title
   panel.setBorder(BorderFactory.createTitledBorder("Risk Analysis Calculator"));

   panel.setLayout(new GridLayout(0,3));
```

```
JPanel leftPanel       = new JPanel(new GridLayout(0,2));
leftPanel.setBorder(BorderFactory.createEmptyBorder(20,25,185,20)); // top, left,
      bottom, right
JPanel middlePanel     = new JPanel(new GridLayout(0,2));
middlePanel.setBorder(BorderFactory.createEmptyBorder(20,0,180,0)); // top, left,
      bottom, right
JPanel rightPanel      = new JPanel(new GridLayout(0,1));
rightPanel.setBorder(BorderFactory.createEmptyBorder(20,25,0,20)); // top, left,
      bottom, right
JPanel rightTopPanel   = new JPanel(new GridLayout(0,1));
JPanel rightBottomPanel = new JPanel(new GridLayout(0,1));
JPanel rightInfoPanel  = new JPanel(new GridLayout(0,1));
rightInfoPanel.setBorder(BorderFactory.createEmptyBorder(25,5,5,5));
JPanel buttonPanel     = new JPanel(new GridLayout(0,2));
buttonPanel.setBorder(BorderFactory.createEmptyBorder(25,5,5,5)); // top, left,
      bottom, right


      rightTopPanel.setBorder(BorderFactory.createTitledBorder(IRV_POTENTIAL_A
      NNUAL_PREMIUM));

      rightBottomPanel.setBorder(BorderFactory.createTitledBorder(IRV_RISK_FACT
      OR));


IRV_ItemNameLabel = new JLabel(IRV_ITEM_NAME);
IRV_ItemNameEntryField = new JFormattedTextField("");
IRV_ItemNameEntryField.addPropertyChangeListener(this);

IRV_PriorityLabel = new JLabel(IRV_PRIORITY);
IRV_PriorityCombo = new JComboBox(HighLowRanges);
IRV_PriorityCombo.setSelectedIndex(HI_LOW_START_INDEX);
IRV_PriorityCombo.addActionListener(this);


IRV_ValueLabel = new JLabel(IRV_VALUE);
IRV_ValueEntryField = new JFormattedTextField("");
IRV_ValueEntryField.addPropertyChangeListener(this);

IRV_LossImpactLabel = new JLabel(IRV_LOSS_IMPACT);
IRV_LossImpactCombo = new JComboBox(HighLowRanges);
IRV_LossImpactCombo.setSelectedIndex(HI_LOW_START_INDEX);
IRV_LossImpactCombo.addActionListener(this);
```

```
IRV_LossExpectancyLabel = new JLabel(IRV_LOSS_EXPECTANCY);
IRV_LossExpectancyCombo = new JComboBox(DateRanges);
IRV_LossExpectancyCombo.setSelectedIndex(DATE_START_INDEX);
IRV_LossExpectancyCombo.addActionListener(this);


IRV_SafeguardCostLabel = new JLabel(IRV_SAFEGUARD_COST);
IRV_SafeguardCostEntryField = new JFormattedTextField("");
IRV_SafeguardCostEntryField.addPropertyChangeListener(this);

IRV_PotentialAnnualPremiumLabel = new
      JLabel(IRV_POTENTIAL_ANNUAL_PREMIUM);
IRV_PotentialAnnualPremiumCalcLabel = new JLabel("");
IRV_PotentialAnnualPremiumCalcLabel.setForeground(Color.blue);


IRV_RiskFactorLabel = new JLabel(IRV_RISK_FACTOR);
IRV_RiskFactorCalcLabel = new JLabel("");
IRV_RiskFactorCalcLabel.setForeground(Color.blue);



leftPanel.add(IRV_ItemNameLabel);
leftPanel.add(IRV_ItemNameEntryField);

leftPanel.add(IRV_ValueLabel);
leftPanel.add(IRV_ValueEntryField);

leftPanel.add(IRV_SafeguardCostLabel);
leftPanel.add(IRV_SafeguardCostEntryField);

middlePanel.add (IRV_PriorityLabel);
middlePanel.add (IRV_PriorityCombo);
middlePanel.add (IRV_LossImpactLabel);
middlePanel.add (IRV_LossImpactCombo);
middlePanel.add (IRV_LossExpectancyLabel);
middlePanel.add (IRV_LossExpectancyCombo);

rightTopPanel.add(IRV_PotentialAnnualPremiumCalcLabel);
rightBottomPanel.add(IRV_RiskFactorCalcLabel);

IRV_InfoLabel = new JLabel("");
rightInfoPanel.add(IRV_InfoLabel);

IRV_CalculateButton = new JButton (IRV_CALCULATE);
```

```
  IRV_CalculateButton.setPreferredSize( new Dimension (10, 5));
  IRV_CalculateButton.addActionListener(this);

  IRV_ClearButton = new JButton (IRV_CLEAR);
  IRV_ClearButton.setPreferredSize( new Dimension (10, 5));
  IRV_ClearButton.addActionListener(this);

  buttonPanel.add(IRV_CalculateButton);
  buttonPanel.add(IRV_ClearButton);

  panel.add (leftPanel);
  panel.add (middlePanel);

  rightPanel.add (rightTopPanel);
  rightPanel.add (rightBottomPanel);
  rightPanel.add (rightInfoPanel);
  rightPanel.add (buttonPanel);
  panel.add (rightPanel);

  return panel;
 }

private JComponent makeItemRiskAssessmentPanel()
{
  JPanel panel = new JPanel(false);

  panel.setLayout(new BorderLayout());
  panel.setBorder(BorderFactory.createTitledBorder("Risk Analysis Calculator"));


  panel.setLayout(new GridLayout (0, 4, 10, 20));

  // Create Natural Disaster Panel populated with checkboxes and entryfield
  JPanel leftPanel   = new JPanel();
  leftPanel.setBorder
        (BorderFactory.createTitledBorder(IRA_NATURAL_DISASTER));
  leftPanel.setLayout(new GridLayout(0,1));


  for (int i = 0; i < NaturalDisasterCheckBoxList.length; i++) {
     NaturalDisasterCheckBoxList[i] = new JCheckBox(NaturalDisasterList[i], false);
     NaturalDisasterCheckBoxList[i].addActionListener(this);
     leftPanel.add(NaturalDisasterCheckBoxList[i]);
  }
```

```
// place holder
leftPanel.add(new JLabel(""));
IRA_ND_MultiplierLabel      = new JLabel (IRA_MULTIPLIER);
IRA_ND_MultiplierEntryField = new JFormattedTextField("");
leftPanel.add(IRA_ND_MultiplierLabel);
leftPanel.add(IRA_ND_MultiplierEntryField);


// Create Accidental Panel populated with checkboxes and entryfield

JPanel middlePanel   = new JPanel();
middlePanel.setBorder (BorderFactory.createTitledBorder(IRA_ACCIDENTAL));
middlePanel.setLayout(new GridLayout(0,1));

for (int i = 0; i < AccidentalCheckBoxList.length; i++) {
   AccidentalCheckBoxList[i] = new JCheckBox(AccidentalList[i], false);
   AccidentalCheckBoxList[i].addActionListener(this);
   middlePanel.add(AccidentalCheckBoxList[i]);
}

middlePanel.add(new JLabel(""));
middlePanel.add(new JLabel(""));
IRA_A_MultiplierLabel      = new JLabel (IRA_MULTIPLIER);
IRA_A_MultiplierEntryField = new JFormattedTextField("");
middlePanel.add(IRA_A_MultiplierLabel);
middlePanel.add(IRA_A_MultiplierEntryField);



// Create Deliberate Panel populated with checkboxes and entryfield

JPanel rightPanel   = new JPanel();
rightPanel.setBorder (BorderFactory.createTitledBorder(IRA_DELIBERATE));
rightPanel.setLayout(new GridLayout(0,1));


for (int i = 0; i < DeliberateCheckBoxList.length; i++) {
   DeliberateCheckBoxList[i] = new JCheckBox(DeliberateList[i], false);
   DeliberateCheckBoxList[i].addActionListener(this);
   rightPanel.add(DeliberateCheckBoxList[i]);
}


IRA_D_MultiplierLabel      = new JLabel (IRA_MULTIPLIER);
IRA_D_MultiplierEntryField = new JFormattedTextField("");
```

```
rightPanel.add(IRA_D_MultiplierLabel);
rightPanel.add(IRA_D_MultiplierEntryField);


JPanel outputPanel      = new JPanel();
outputPanel.setLayout(new GridLayout(0,1));
JPanel outputNDPanel    = new JPanel();
JPanel outputAPanel     = new JPanel();
JPanel outputDPanel     = new JPanel();
JPanel outputSIRVPanel  = new JPanel();
JPanel outputInfoPanel  = new JPanel();


    outputNDPanel.setBorder(BorderFactory.createTitledBorder(IRA_ND_RISK_VA
    LUATION));

    outputAPanel.setBorder(BorderFactory.createTitledBorder(IRA_A_RISK_VALU
    ATION));

    outputDPanel.setBorder(BorderFactory.createTitledBorder(IRA_D_RISK_VALU
    ATION));

    outputSIRVPanel.setBorder(BorderFactory.createTitledBorder(IRA_SINGLE_IT
    EM_RISK_VALUATION));


IRA_ND_ValuationCalcLabel = new JLabel ("");
outputNDPanel.add(IRA_ND_ValuationCalcLabel);

IRA_A_ValuationCalcLabel  = new JLabel ("");
outputAPanel.add(IRA_A_ValuationCalcLabel);

IRA_D_ValuationCalcLabel  = new JLabel ("");
outputDPanel.add(IRA_D_ValuationCalcLabel);

IRA_SingleItemRiskValuationCalcLabel = new JLabel("");
outputSIRVPanel.add(IRA_SingleItemRiskValuationCalcLabel);

IRA_InfoLabel = new JLabel("");
outputInfoPanel.add (IRA_InfoLabel);


IRA_CalculateButton = new JButton(IRA_CALCULATE);
IRA_CalculateButton.addActionListener(this);
```

```
  IRA_ClearButton = new JButton(IRA_CLEAR);
  IRA_ClearButton.addActionListener(this);

  JPanel buttonPanel = new JPanel(new GridLayout(0,2));
  buttonPanel.add (IRA_CalculateButton);
  buttonPanel.add (IRA_ClearButton);

  outputPanel.add (outputNDPanel);
  outputPanel.add (outputAPanel);
  outputPanel.add (outputDPanel);
  outputPanel.add (outputSIRVPanel);
  outputPanel.add (outputInfoPanel);
  outputPanel.add (buttonPanel);


  panel.add (leftPanel);
  panel.add (middlePanel);
  panel.add (rightPanel);
  panel.add (outputPanel);


  return panel;
}



/**
 * Create the GUI and show it. For thread safety, this method should be
 * invoked from the event-dispatching thread.
 */
private static void createAndShowGUI()
{
  // Make sure we have nice window decorations.
  JFrame.setDefaultLookAndFeelDecorated(true);

  // Create and set up the window.
  JFrame frame = new JFrame("Risk Analysis");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  // Create and set up the content pane.
  JComponent newContentPane = new RiskAnalysis();
  newContentPane.setOpaque(true); // content panes must be opaque

  frame.getContentPane().add(newContentPane, BorderLayout.CENTER);
```

```
    // Display the window.
    frame.pack();
    frame.setVisible(true);
  }

  public static void main(String[] args)
  {
    // Schedule a job for the event-dispatching thread:
    // creating and showing this application's GUI.
    javax.swing.SwingUtilities.invokeLater(new Runnable()
    {
      public void run()
      {
        createAndShowGUI();
      }
    });
  }
}
```