REPORT ON POSSIBLE SOLUTIONS FOR VARCHART XGANTT CRASHING ISSUE

1 PROBLEM DEFINITION

After filling in details on the "Prepare Work Order" form in the IFS application, right clicking and selecting "Allocate Resource" creates a dialog where a Gantt chart can be configured. There are no errors until the resource is dragged and dropped to the lower section of the window. However after the drag-and-drop, clicking anywhere on the upper window will bring up an error message, and the system will crash. The main issue with the Netronic error that appeared was that there was no way to get further details as neither the website nor the manual contained meanings of error codes or classes generated by VarChart xGantt.

2 ERROR

Clicking on the upper area of the window results in the occurrence of an unhandled "System.AccessViolationException", which by definition is the "exception that is thrown when there is an attempt to read from or write to protected memory" [1]. The complete error message is included in the appendix. The reason for this error is given as "Form that is already visible cannot be displayed as a modal dialog box. Set the form's visible property to false before calling showDialog."

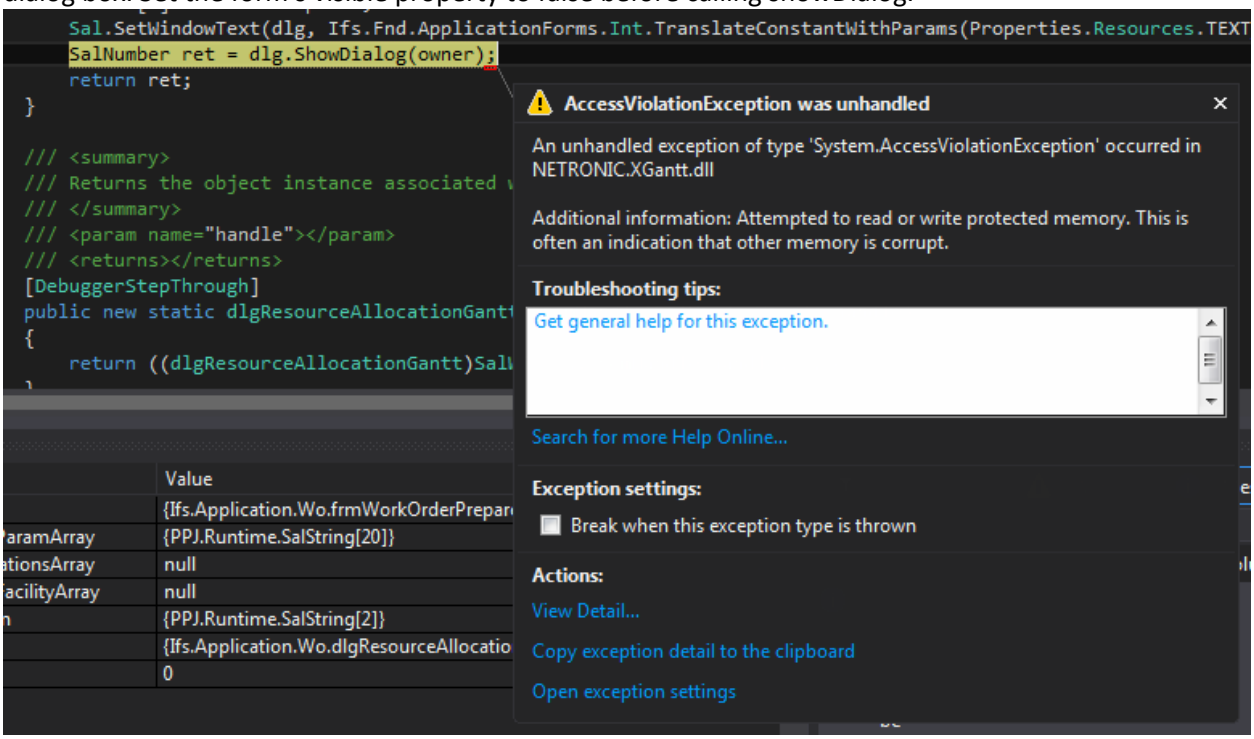

Figure 1-Exception Details

3 FINDINGS

After adding breakpoints, observing the code, and searching online forums, several possible solutions were found and tried out;

3.1 NETRONIC PROPERTIES

After referring to the manual for Netronic Xgantt Varchart for .Net 5.0, changing the properties seemed like a good way to begin. Section 4 of the manual gives details on property pages and dialog boxes, and was a useful reference. Changing the different options on the properties menu and trying out different combinations yielded no result, although some combinations resulted in different error messages than the usual(for example trying to set the upper section's visibility as "false" after the drag-and-drop event resulted in error code 152 rather than 149).

3.2 TRY/CATCH BLOCK

The next option tried out was to encapsulate the statement causing the error within a basic try/catch block. However, various forums on MSDN [2], Stack Overflow [3] [4]and CodeProject [5] pointed out rather than using exception blocks [6], it is more effective to have one error handler to handle all unhandled exceptions, by using one of .NET's unhandled exception event handlers. This was done by creating a separate thread for exception handling. After adding the code, there was some difficulty in building the code as Visual Studio itself crashed. However after successfully building there was no change in the error generated by Netronic xGantt and the system still froze after "Error 4 Class 149". All code added was then removed and the app code was reloaded.

3.3 WIN32 ERROR HANDLING

Since it did not appear to be possible to use a basic try/catch statement or event handlers, Win32 error handling was tried out. The third party (the Netronic application) creating an unhandled and unspecified error which gets transferred to the main thread as a System.Event causes the crash. One of the options on some other Stack Overflow forums [7] [8] was to use Win32 exception handling [9] instead. Accordingly the exception in the vcGantt_Click was changed to a Win32Exception. This would not build successfully; an error message appeared that there is no such overload for vcGantt_Click.

3.4 CHANGING THE .NET CONFIGURATION FILE

Many answers on Stack Overflow point out that there is no way to catch exceptions that occur on threads of external applications or components without correcting the root cause of the error itself [10], an option which is not available as Netronic is not a FOSS.

However these forums say that there is a workaround; to change the .NET configuration file of versions 2.0 and later to the exception handling method used by earlier versions of .NET [11] [12]. This solution simply consists of changing the runtime configuration located in the .NET framework folder and setting `legacyUnhandledExceptionPolicy enabled="true"`.

Unfortunately, this would not be a viable solution as implementing it would require that all config files of users of the system be changed. This could (hypothetically) be done from within the code, opening the config file, changing the required values and then restoring it to its previous state [13]. However the Microsoft Support forum that provides this solution [12] contains a warning that "We do not recommend that you change the default behavior. If you ignore exceptions, the application may leak resources and abandon locks."

3.5 Setting the Diagram's Visibility to False After the Node is Moved

According to section 3.5 of the VarChart xGantt manual, the drag and drop action is done in two phases. The source component and target component each have separate events which are triggered. Setting the "diagramVisible" property of the upper diagram to false once one of these events are triggered should keep the click event from being recognized, and stop the system crash. After adding a Boolean flag within the Control.GiveFeedback function, and a breakpoint it was found that the flag changes as soon as the node in the upper layer is clicked on, but not for any of the nodes in the lower layer.

The code was edited to disable the visibility of the diagram as soon as the node was dragged away from it. A Boolean variable "trig" was added to restore the visibility in case "undo" was clicked on. This way, since the user can't click on a faulty section once the drag-and-drop has been completed, the system crash is avoided.

New code added:

```
private void vcJobGantt_GiveFeedback(object sender, GiveFeedbackEventArgs e)
        {
            trig = true;
            vcJobGantt.Enabled = false;

        }
```

In case of multiple operations, it is necessary to have the diagram visible until all nodes are moved. The trig variable is used here to check whether at least one node has been moved. If it has been moved already and the new node is for another operation, then the diagram will become visible again, until Control.GiveFeedback is triggered again.

However there is an issue in that changing the code disabled the "save" function as well. This error was corrected for the work order allocation, but accessing through the operation tab and selecting multiple "planned men" creates an issue as the diagram remains disabled. Enabling the diagram in this instance also enables the click event being recognized, and creates another "Error 4" from Netronic.

3.6 Enabling/Disabling the Diagram

Since changing the visibility made the diagram disappear completely, disabling it was tried out. The Netronic error only appears if the area is clicked on after the node has been moved for the first time. Therefore disabling and then enabling the diagram after each drag and drop operation of a node would make the appearance of the error unnecessary.

Final modification made to the code :

```
private void vcJobGantt_GiveFeedback(object sender, GiveFeedbackEventArgs e)
        {
            vcJobGantt.Enabled = false;
            vcJobGantt.Enabled = true;
        }
```

Reloading the work order folder and adding the enabling/disabling code again stopped the save error as well. This was chosen as a viable solution.

4 Appendix

4.1 References

[1] Microsoft, "Troubleshooting Exceptions: System.AccessViolationException," [Online]. Available: https://msdn.microsoft.com/en-us/library/ms164911.aspx. [Accessed 08 2015].

[2] A. Pardoe, "Handling Corrupted State Exceptions," [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dd419661.aspx. [Accessed 08 2015].

[3] D. Young, "Stack Overflow- Catching Unhandled Exceptions on Seperate Threads," [Online]. Available: http://stackoverflow.com/questions/4284986/catching-unhandled-exception-on-separate-threads. [Accessed 08 2015].

[4] M. Caron, "Stack Overflow- Finding the Cause of System.AccessViolationException," [Online]. Available: http://stackoverflow.com/questions/5133971/finding-the-cause-of-system-accessviolationexception. [Accessed 08 2015].

[5] K. McFarlene, "Code Project - Managing Unhandled Exceptions in .NET," [Online]. Available: http://www.codeproject.com/Articles/2949/Managing-Unhandled-Exceptions-in-NET. [Accessed 08 2015].

[6] Microsoft, "Exceptions and Exception Handling - C# Programming Guide," [Online]. Available: https://msdn.microsoft.com/en-us/library/ms173160.aspx. [Accessed 08 2015].

[7] D. Khan, "Stack Overflow- Prevent Unwanted Close of Program on Crash," [Online]. Available: http://stackoverflow.com/questions/21160149/prevent-unwanted-close-of-program-on-crash-including-with-try-catch. [Accessed 08 2015].

[8] Vivek, "Stack Overflow - Catching a Native Exception in C#," [Online]. Available: http://stackoverflow.com/questions/150544/can-you-catch-a-native-exception-in-c-sharp-code. [Accessed 08 2015].

[9] Microsoft, "MSDN- Win32Exception Class," [Online]. Available: https://msdn.microsoft.com/en-us/library/system.componentmodel.win32exception.aspx. [Accessed 08 2015].

[10] J. Saunders, "Stack Overflow- Preventing Exceptions from a 3rd Party Component Crashing the Entire Application," [Online]. Available: http://stackoverflow.com/questions/6157696/preventing-exceptions-from-3rd-party-component-from-crashing-the-entire-applicat. [Accessed 08 2015].

[11] Y. Schwarz, "Stack Overflow- How to Prevent an Exception in a Background Thread from Terminating an Application," [Online]. Available: http://stackoverflow.com/questions/186854/how-to-prevent-an-exception-in-a-background-thread-from-terminating-an-applicatio. [Accessed 08 2015].

[12] Microsoft, "Unhandled Exceptions Cause Applications to Quit Unexpectedly in the .NET Framework," [Online]. Available: https://support.microsoft.com/en-us/kb/911816. [Accessed 08 2015].

[13] J. Saunders, "Stack Overflow- App.Config Change Value in C#," [Online]. Available: http://stackoverflow.com/questions/11149556/app-config-change-value. [Accessed 08 2015].

[14] Microsoft, "InvalidOperationException Class - MS Developer Network," [Online]. Available: https://msdn.microsoft.com/en-us/library/system.invalidoperationexception(v=vs.110).aspx. [Accessed 8 2015].

4.2 UNHANDLED EXCEPTION DETAILS

System.InvalidOperationException was unhandled by user code
HResult=-2146233079
Message=Form that is already visible cannot be displayed as a modal dialog box. Set the form's visible property to false before calling showDialog.
Source=System.Windows.Forms
StackTrace:
at System.Windows.Forms.Form.ShowDialog(IWin32Window owner)
at System.Windows.Forms.Form.ShowDialog()
at PPJ.Runtime.Windows.SalForm.ShowDialog(Control parent)
at Ifs.Fnd.ApplicationForms.cDialog.ShowDialog(Control parent)
at Ifs.Application.Wo.dlgResourceAllocationGantt.ModalDialog(Control owner, SalArray`1 sDataParamArray, SalArray`1 sOperationsArray, SalArray`1 sToolFacilityArray)
at Ifs.Application.Wo.frmWorkOrderPrepareTab.menuFrmMethods_ResoAllAssitance_Execute(Object sender, FndCommandExecuteEventArgs e)
at Ifs.Fnd.Windows.Forms.FndCommand.OnExecute(Component component)
at Ifs.Fnd.Windows.Forms.FndContextMenuStrip.ExecuteCommand(ToolStripMenuItem menuItem, EventArgs e)
at Ifs.Fnd.ApplicationForms.FndContextMenuStripInternal.ExecuteCommand(ToolStripMenuItem menuItem, EventArgs e)
at Ifs.Fnd.Windows.Forms.FndToolStripMenuItem.OnClick(EventArgs e)
at System.Windows.Forms.ToolStripItem.HandleClick(EventArgs e)
at System.Windows.Forms.ToolStripItem.HandleMouseUp(MouseEventArgs e)
at System.Windows.Forms.ToolStripItem.FireEventInteractive(EventArgs e, ToolStripItemEventType met)
at System.Windows.Forms.ToolStripItem.FireEvent(EventArgs e, ToolStripItemEventType met)

at System.Windows.Forms.ToolStrip.OnMouseUp(MouseEventArgs mea)
at System.Windows.Forms.ToolStripDropDown.OnMouseUp(MouseEventArgs mea)
at System.Windows.Forms.Control.WmMouseUp(Message& m, MouseButtons button, Int32 clicks)
at System.Windows.Forms.Control.WndProc(Message& m)
at System.Windows.Forms.ScrollableControl.WndProc(Message& m)
at System.Windows.Forms.ToolStrip.WndProc(Message& m)
at System.Windows.Forms.ToolStripDropDown.WndProc(Message& m)
at System.Windows.Forms.Control.ControlNativeWindow.OnMessage(Message& m)
at System.Windows.Forms.Control.ControlNativeWindow.WndProc(Message& m)
at System.Windows.Forms.NativeWindow.Callback(IntPtr hWnd, Int32 msg, IntPtr wparam, IntPtr lparam)
InnerException