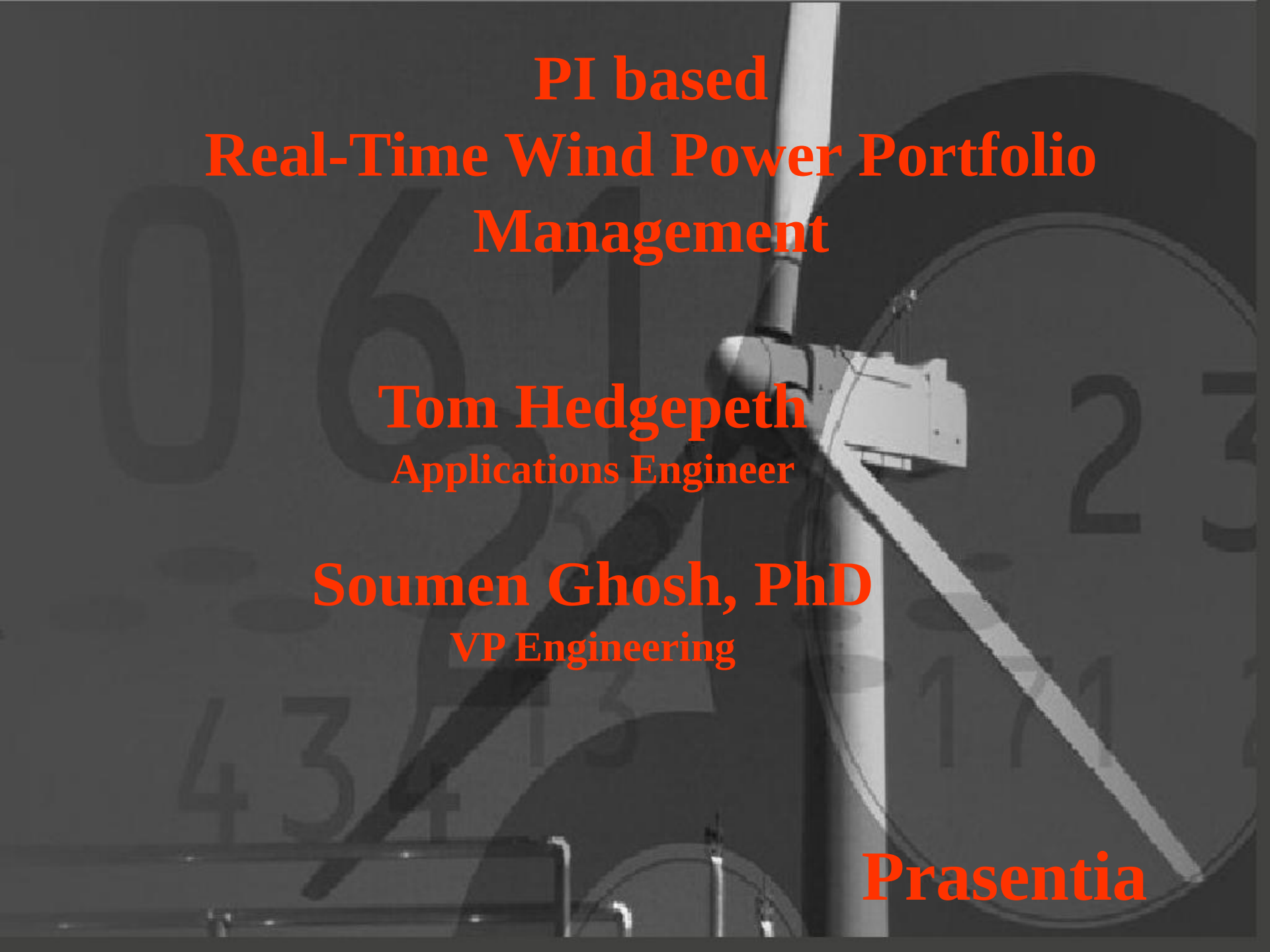


# PI based Real-Time Wind Power Portfolio Management


**Tom Hedgepeth**
Applications Engineer

**Soumen Ghosh, PhD**
VP Engineering

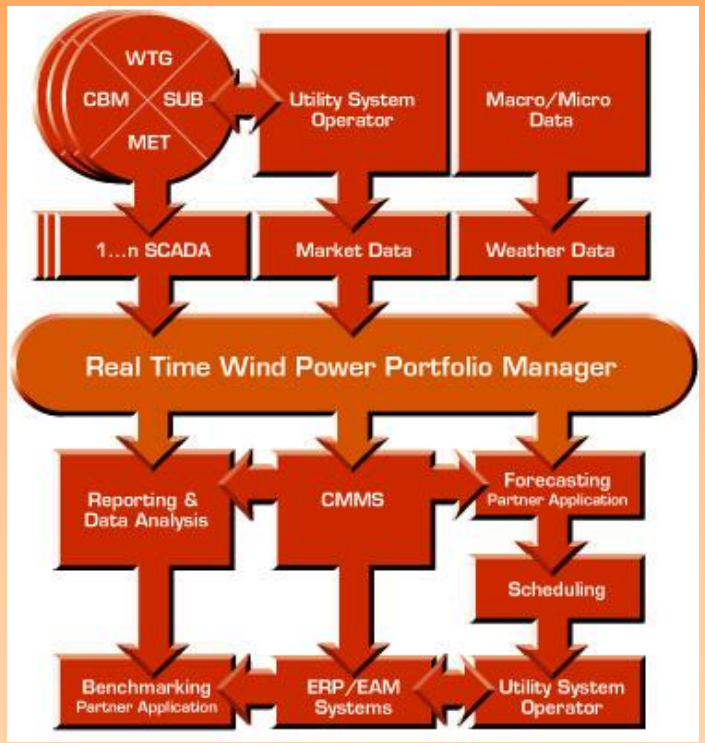
**Prasentia**

# Agenda

- About Prasentia
- Current Configuration
- Standardizing Multiple Turbine Types
- Creating Faults – Unit Batches
- Using Visual Studio, Crystal Reports, and the PI-SDK
- Lessons Learned
- Questions

# About Prasentia

- Best of breed technology integrator specializing in Wind Power.
- SCADA:
  - Siemens Simatic IT Historian
  - Communication Directly from SCADA to WTG Controller, Meteorological Station, and Substation Controllers
  - OPC Compliant
- Reporting and Data Analysis:
  - OSISoft PI Historian
  - Visual Studio .Net 2003
    - Crystal Reports
    - ASP.Net
    - C#
- Visualization Tools:
  - OSISoft ICE Platform
  - ProcessBook and DataLink



# Previous Wind Farm Management Tools

- An off-line information and reporting service attached to individual wind farm
- Based on home-grown database, (Microsoft Access, SQL, other). Required large data transfers to perform portfolio analysis.
- Reports available around the tenth or twelfth of the month
- Not timely enough to give decision makers the advantage to proactively steer the project in the most optimal directions.

# Genesis of RT-WPPM

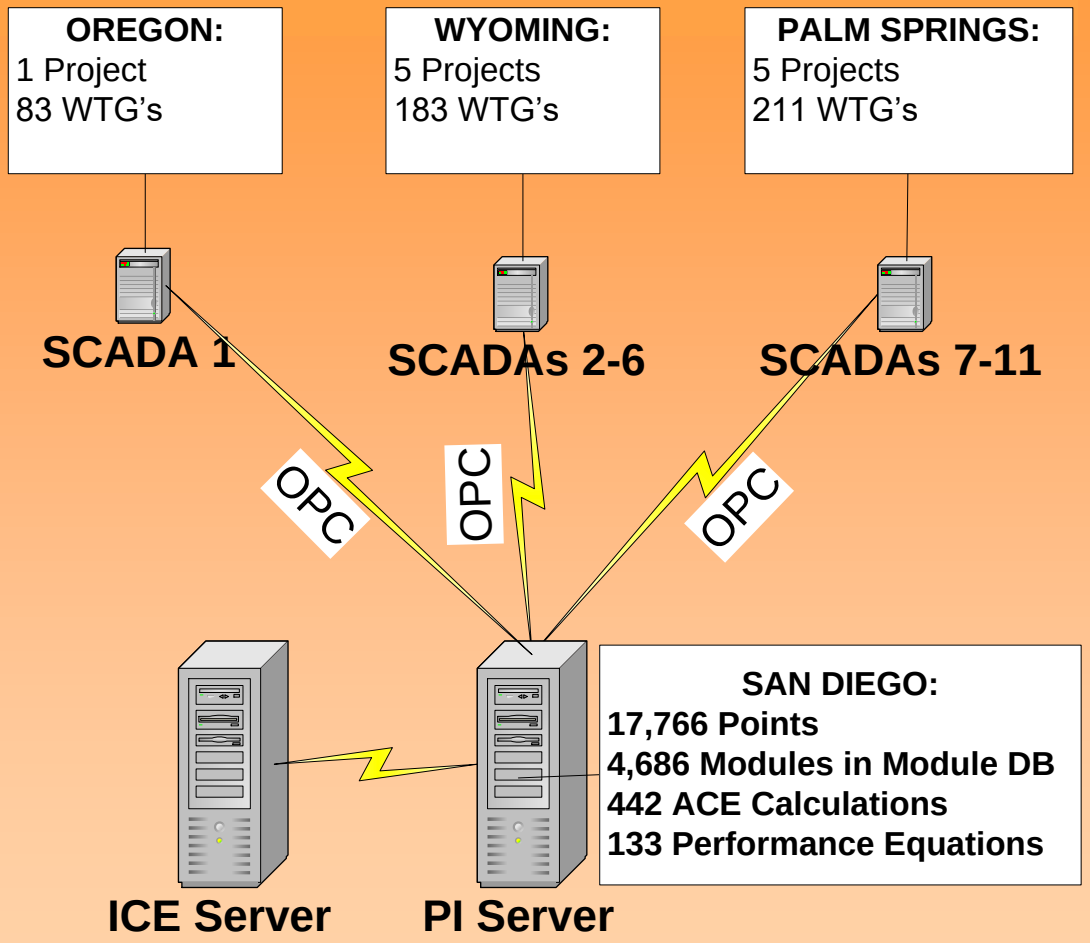
- To provide wind industry with
  - portfolio based  strategic information in real-time
  - applications to optimize wind assets and performance

- To help stakeholders
  - maximize the economics of wind projects by delivering critical information in real-time from the **same data** source to every level

# Current Configuration

WTG Models

- Micon
  - 48/700
  - 750
- Mitsubishi
  - MWT 600
  - MWT 450/600
  - MWT 1000
  - MWT 600-47
- Vestas
  - V47

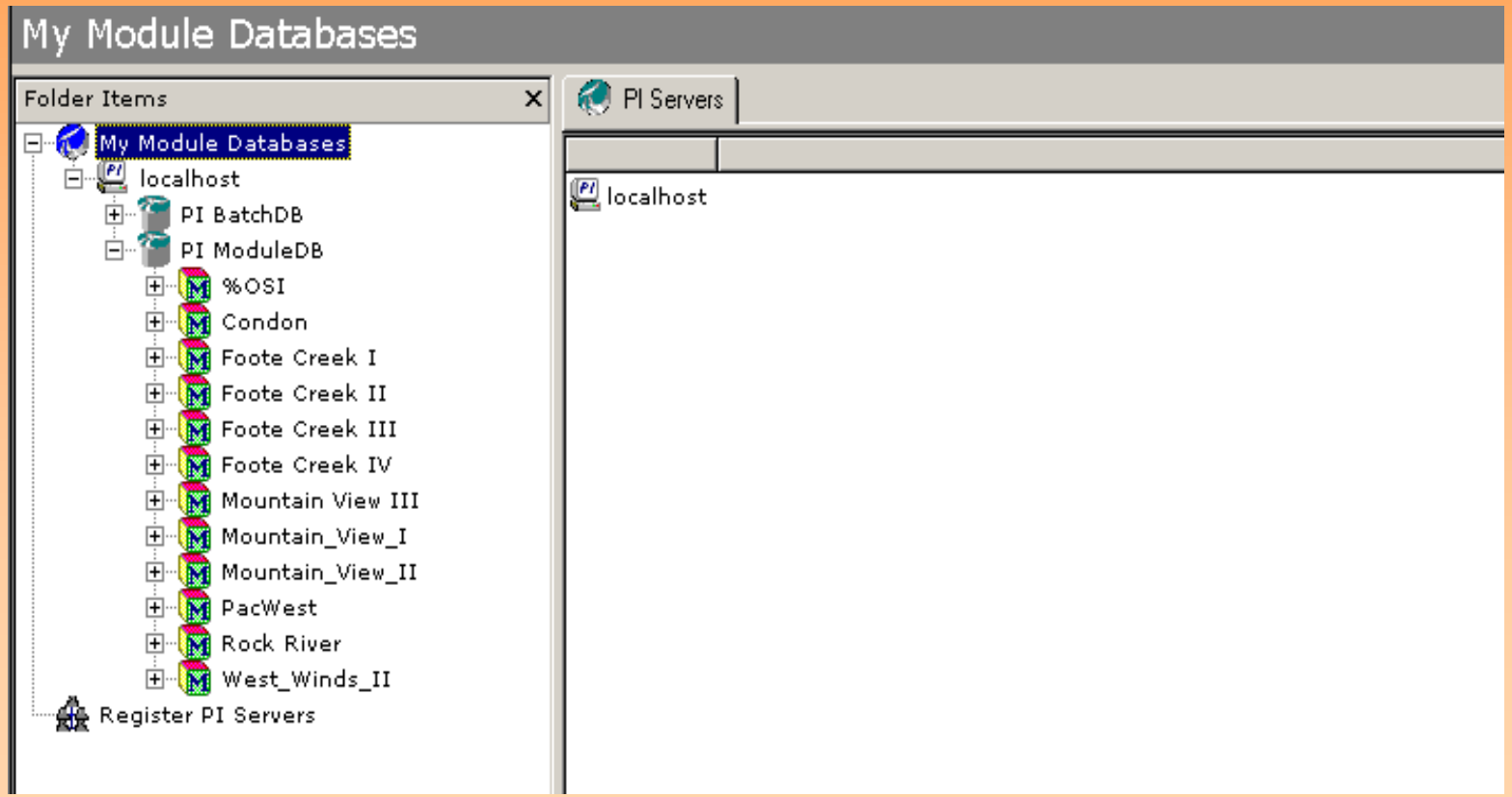


# Standardizing Multiple Turbine Types

- **Goal: Provide a user friendly "Alias" to cryptic point names while standardizing multiple vendors for reporting purposes.**
- Facts:
  - All wind turbines have the same basic components.
  - Reporting requirements generally require the same basic inputs.

# Standardizing Multiple Turbine Types

- **Solution: OsiSoft Module Database!**



# WHY PI Module Data Base

- Nature of Wind Farm has a consistent hierarchy
  - Wind Farm (Project)
    - Wind Turbine Generator
    - Substation
    - Meteorological Station
- Prasentia applications use reusable elements to map PI Data for different user requirements
  - Operator, Owner, Performance Engr, Turbine Manufacturer
- Efficient access to Tags using Aliases for PI data sources.
- Manage additions and changes to the relationship of Wind Farm data over time
- Use Batch processes to track turbine status and faults

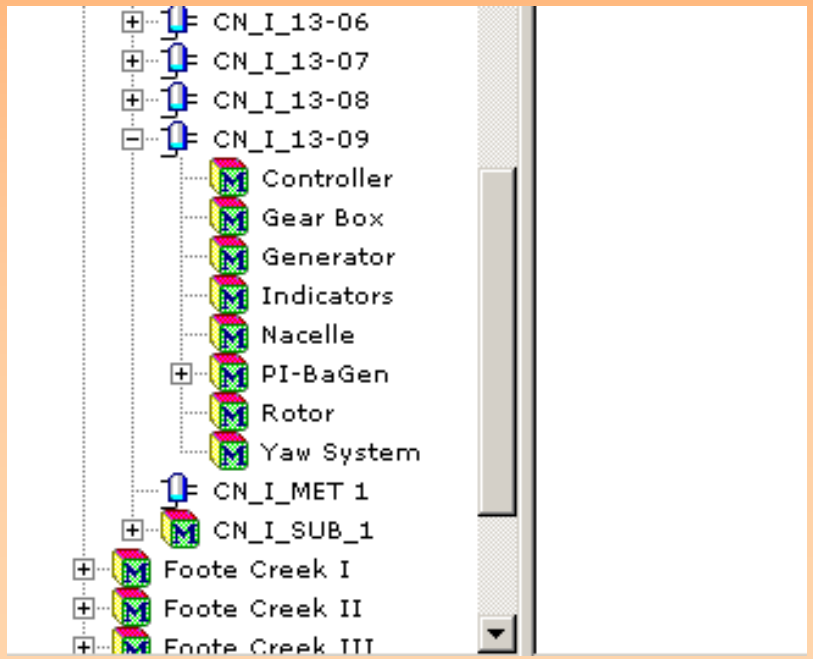
# Our Module Database Structure

- Project 1
  - WTG 1
    - Controller
    - Nacelle
    - etc
  - WTG 2
  - Substation
  - Met 1
- Project 2

Has Properties

Has Aliases

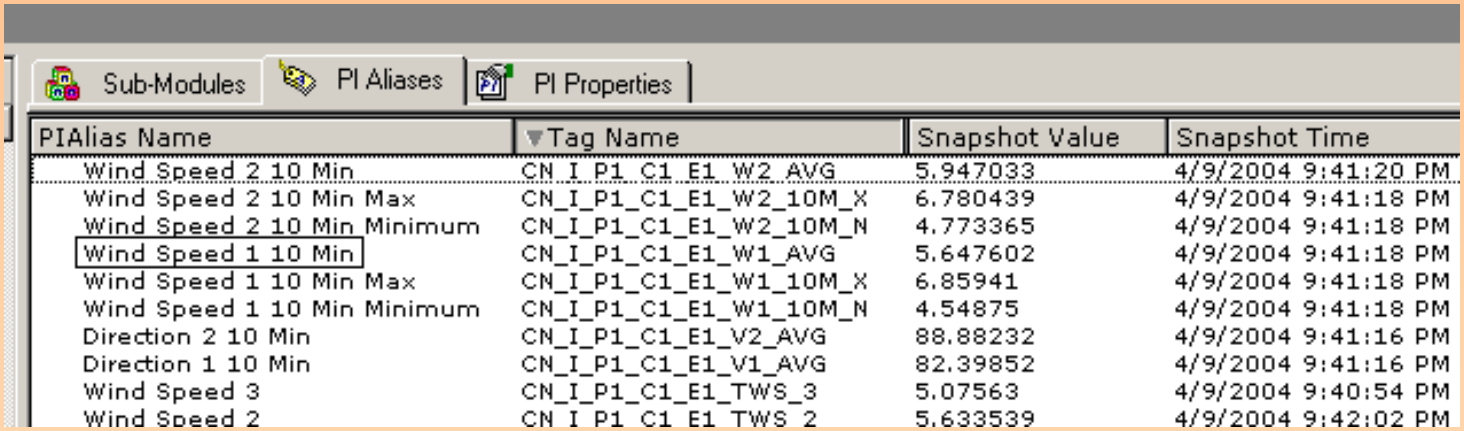
Is PiUnit




Properties?
Aliases?
PiUnit?

# Module Database & Batch Terminology

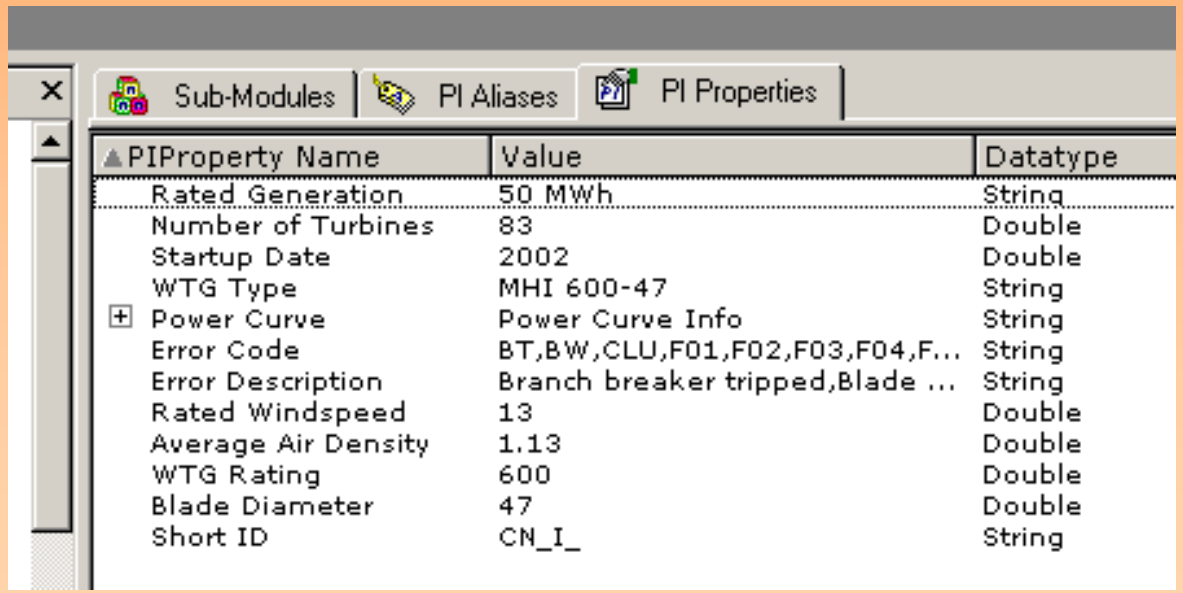
- Alias: “Aliases provide a translation from site-specific plant instrument tags to more generic, process-oriented names, which are more intuitive to users” (OSISoft Devnet).
  - CN_I_P1_C1_E1_TWS1 = Project \ Met 1 \ Wind Speed 1
  - CN_I_P1_C1_M001_VGEN = Project \ WTG 1 \ Generator \ RPM




| PIAlias Name | Tag Name | Snapshot Value | Snapshot Time |
|---|---|---|---|
| Wind Speed 2 10 Min | CN_I_P1_C1_E1_W2_AVG | 5.947033 | 4/9/2004 9:41:20 PM |
| Wind Speed 2 10 Min Max | CN_I_P1_C1_E1_W2_10M_X | 6.780439 | 4/9/2004 9:41:18 PM |
| Wind Speed 2 10 Min Minimum | CN_I_P1_C1_E1_W2_10M_N | 4.773365 | 4/9/2004 9:41:18 PM |
| Wind Speed 1 10 Min | CN_I_P1_C1_E1_W1_AVG | 5.647602 | 4/9/2004 9:41:18 PM |
| Wind Speed 1 10 Min Max | CN_I_P1_C1_E1_W1_10M_X | 6.85941 | 4/9/2004 9:41:18 PM |
| Wind Speed 1 10 Min Minimum | CN_I_P1_C1_E1_W1_10M_N | 4.54875 | 4/9/2004 9:41:18 PM |
| Direction 2 10 Min | CN_I_P1_C1_E1_V2_AVG | 88.88232 | 4/9/2004 9:41:16 PM |
| Direction 1 10 Min | CN_I_P1_C1_E1_V1_AVG | 82.39852 | 4/9/2004 9:41:16 PM |
| Wind Speed 3 | CN_I_P1_C1_E1_TWS_3 | 5.07563 | 4/9/2004 9:40:54 PM |
| Wind Speed 2 | CN_I_P1_C1_E1_TWS_2 | 5.633539 | 4/9/2004 9:42:02 PM |

# Other Terms

- Property: “Used to store static information about a module, like equipment specifications” (OSISoft Devnet)
- PIUnit: Classifies the Module as a Batch Unit.

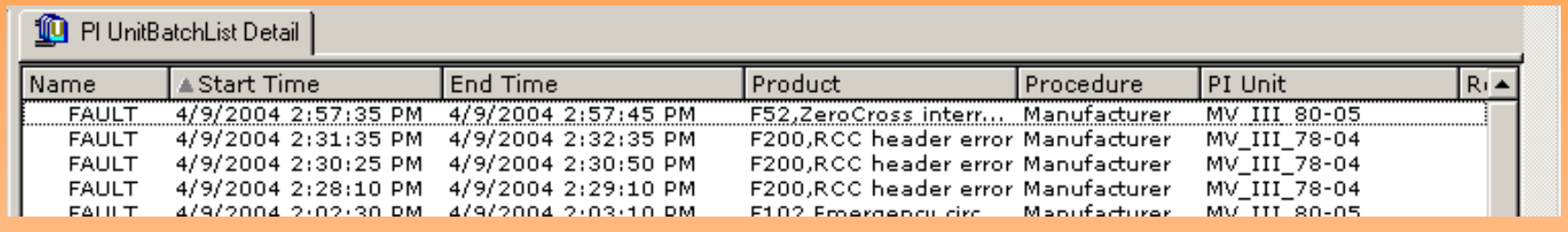


# Using Unit Batches to track Down Time

- PI Batch Generator Interface:
  - Used to automatically create UnitBatches and SubBatches.
  - Monitors 1 active point to start/stop UnitBatches.
  - Allows assignment of tags for associating values to the Product and Procedure fields.

# Using Unit Batches to track Down Time

- Prasentia's Standard Configuration:

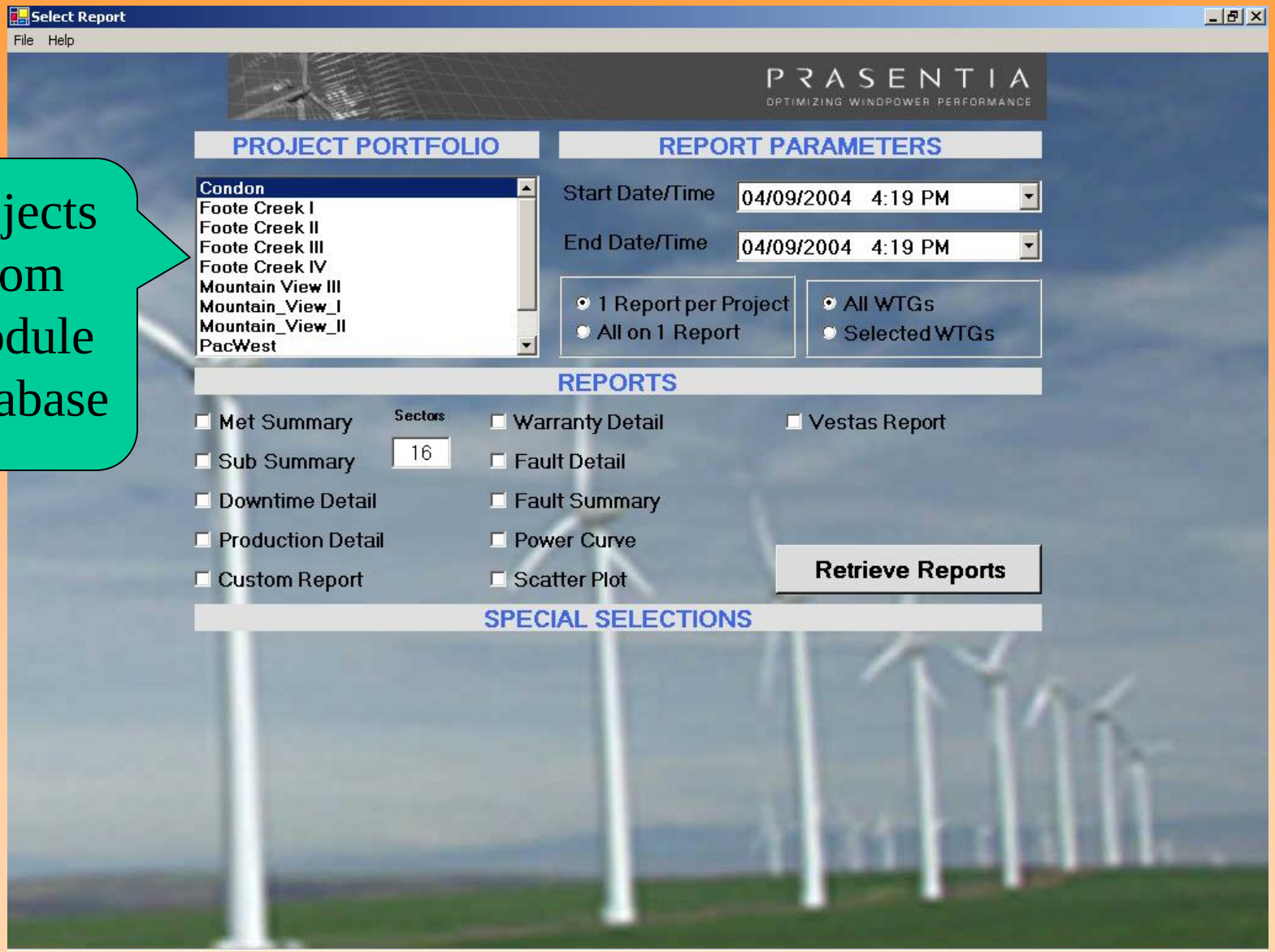

PI UnitBatchList Detail

| Name | Start Time | End Time | Product | Procedure | PI Unit | R |
|---|---|---|---|---|---|---|
| FAULT | 4/9/2004 2:57:35 PM | 4/9/2004 2:57:45 PM | F52,ZeroCross interr... | Manufacturer | MV_III_80-05 | |
| FAULT | 4/9/2004 2:31:35 PM | 4/9/2004 2:32:35 PM | F200,RCC header error | Manufacturer | MV_III_78-04 | |
| FAULT | 4/9/2004 2:30:25 PM | 4/9/2004 2:30:50 PM | F200,RCC header error | Manufacturer | MV_III_78-04 | |
| FAULT | 4/9/2004 2:28:10 PM | 4/9/2004 2:29:10 PM | F200,RCC header error | Manufacturer | MV_III_78-04 | |
| FAULT | 4/9/2004 2:02:30 PM | 4/9/2004 2:03:10 PM | F102,Emergency circ... | Manufacturer | MV_III_80-05 | |

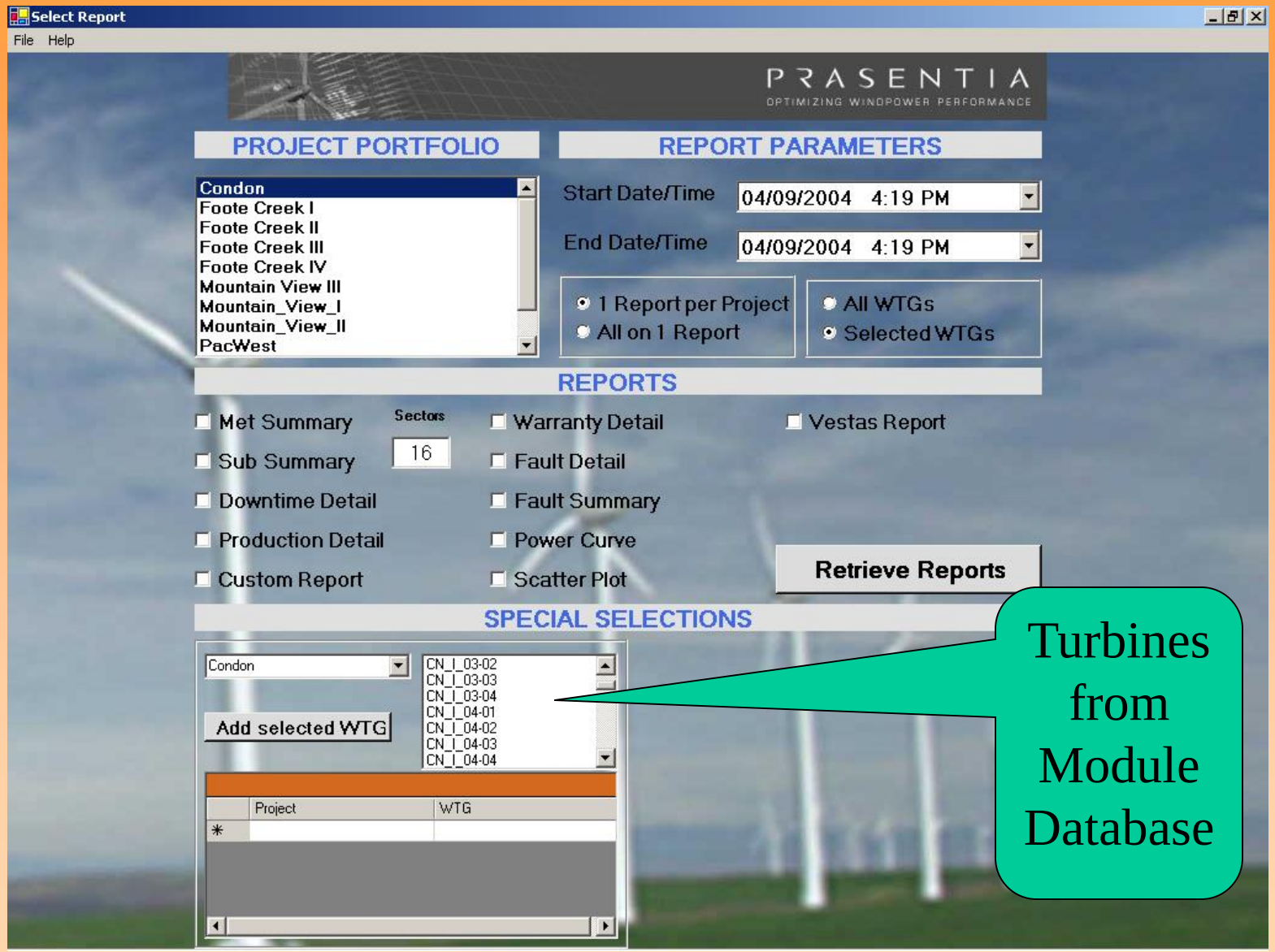
- WTG Status Point is used as the "Active Point" and "Batch ID".
- Procedure is standardized as a Downtime Category.
- Product is used for a description of the fault.

# Visual Studio 2003

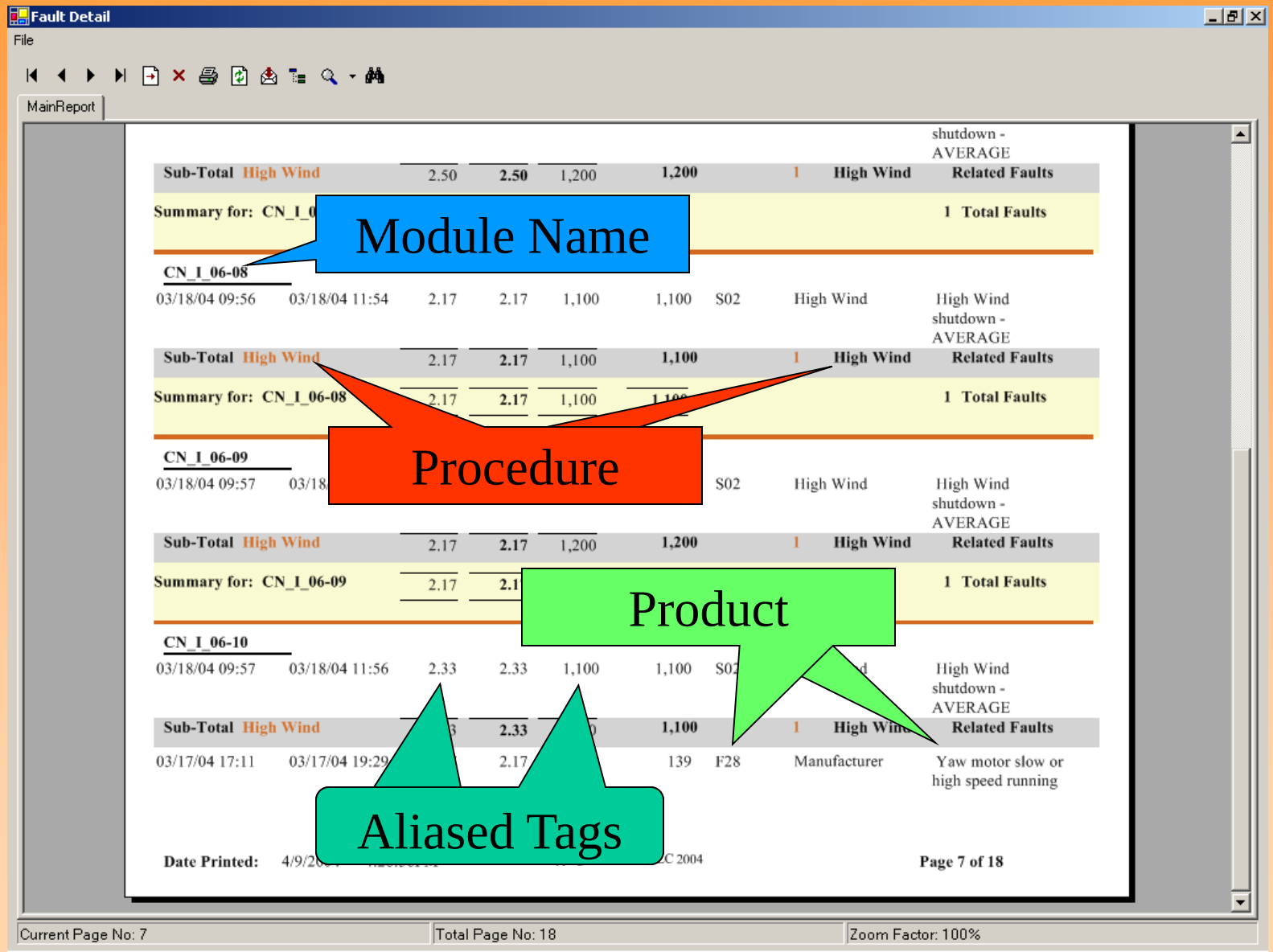
Projects from Module Database



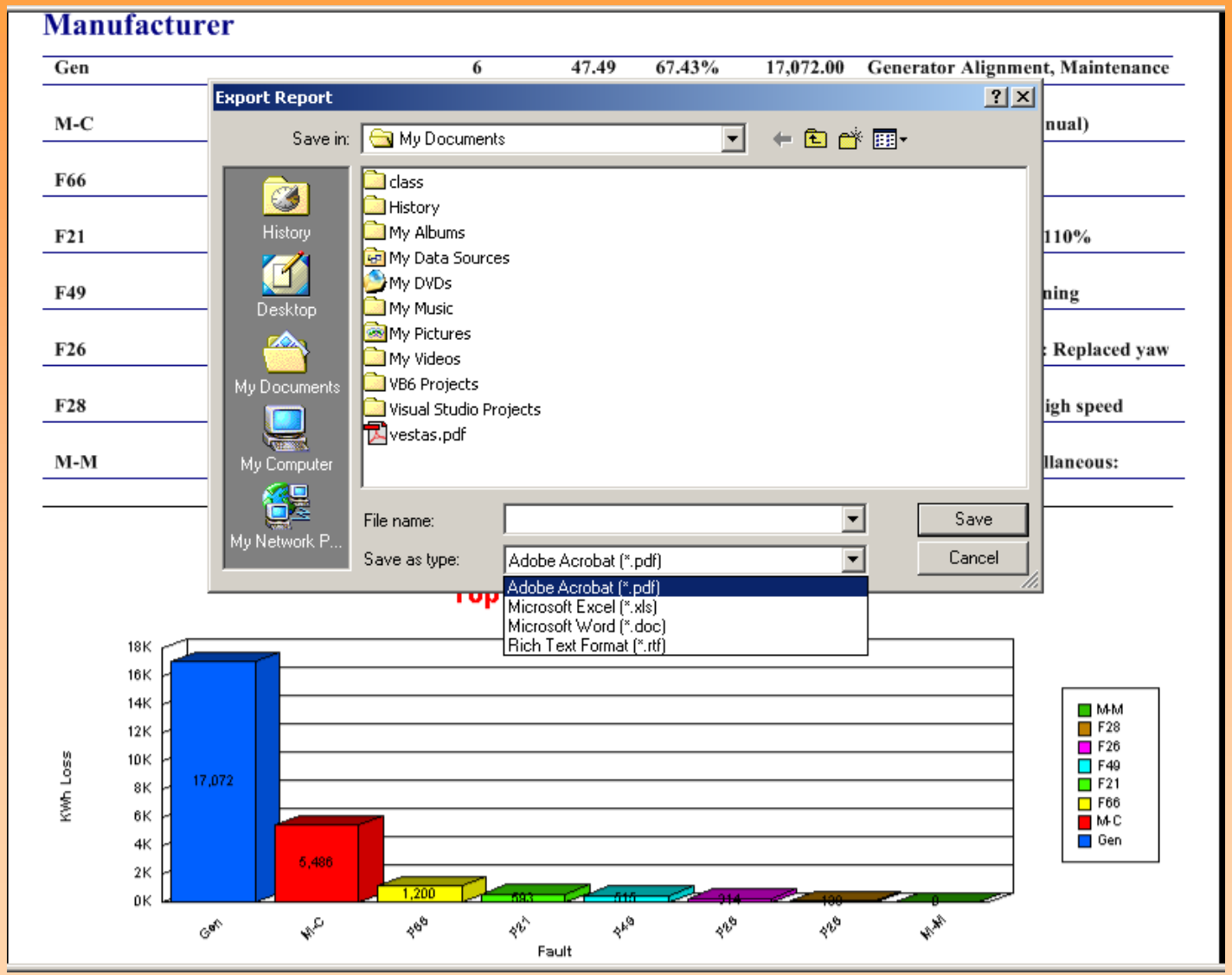
# Visual Studio 2003



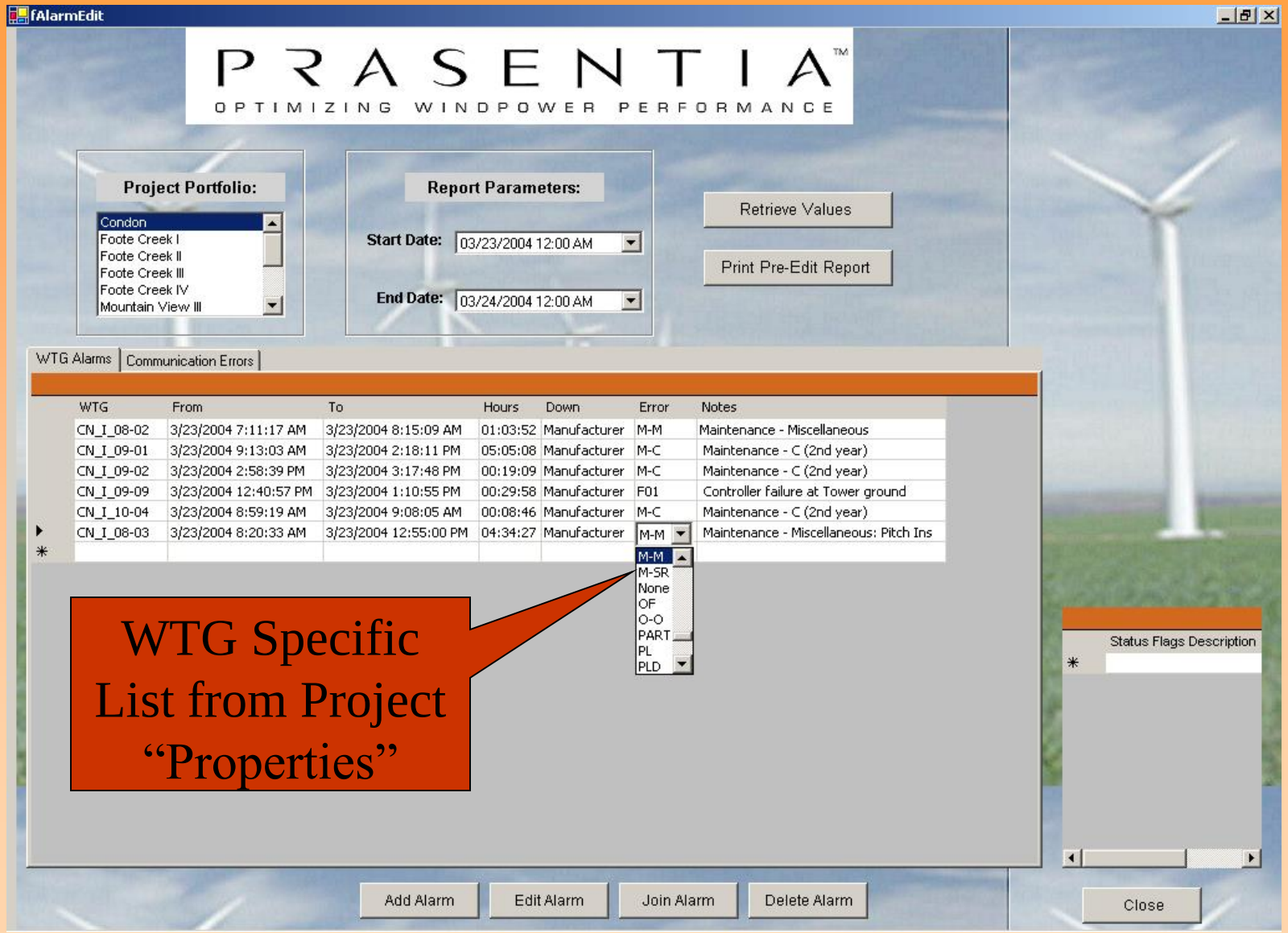
# Crystal Reports



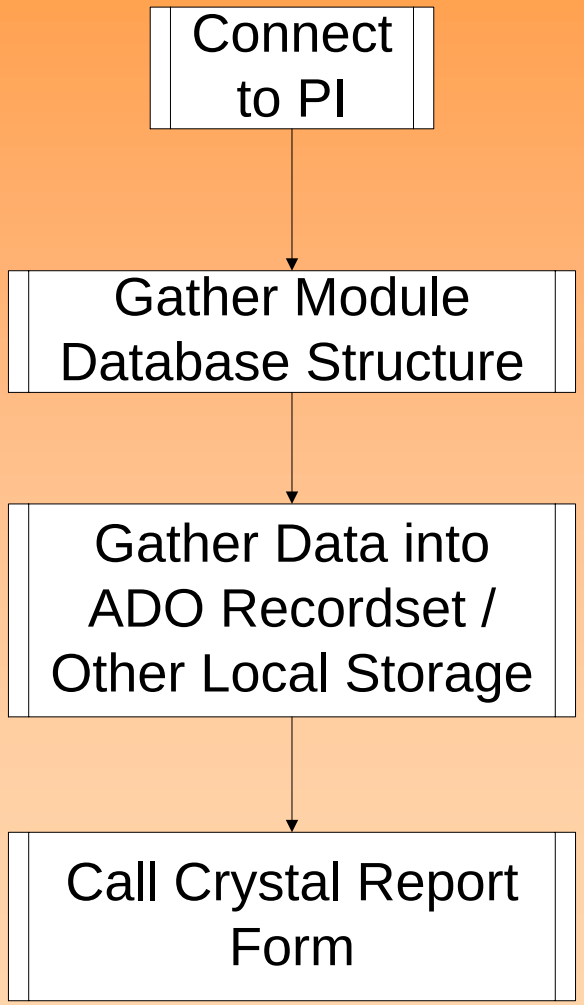
# Crystal Reports



# Editing Turbine Faults



# Visual Studio 2003 in a Nutshell




# Visual Studio 2003: How To

Connect to PI

```
PISDK.PISDK mySDK;
mySDK = new PISDK.PISDKClass();
g_PIServer = mySDK.Servers[strServer];

if (!(g_PIServer.Connected))
{
   PISDKDlg.Connections cnxn = new PISDKDlg.ConnectionsClass();
   cnxn.Login(ref g_PIServer,ref strUN,ref strPW, true, false);
}
```

# Visual Studio 2003: How To

Gather Module
Database Structure

1. Create a PIModule “WarrModule”
2. Set WarrModule equal to the desired project in the Database
3. Create a PIProperty “propRating”
4. Set a PIProperty equal to the desired Property

```
PISDK.PIModule WarrModule;
WarrModule = g_PIServer.PIModuleDB.PIModules[strProject];
PISDK.PIProperty propRating;
propRating = WarrModule.PIProperties["WTG Rating"];
```

# Visual Studio 2003: How To

Gather Data into ADO Recordset / Other Local Storage

1. Set the “GeneratorMod” to a Turbines sub module.
2. Retrieve the desired Alias
3. Retrieve the point associated with the alias
4. Retrieve the data from the archive, this could also be a call for multiple values

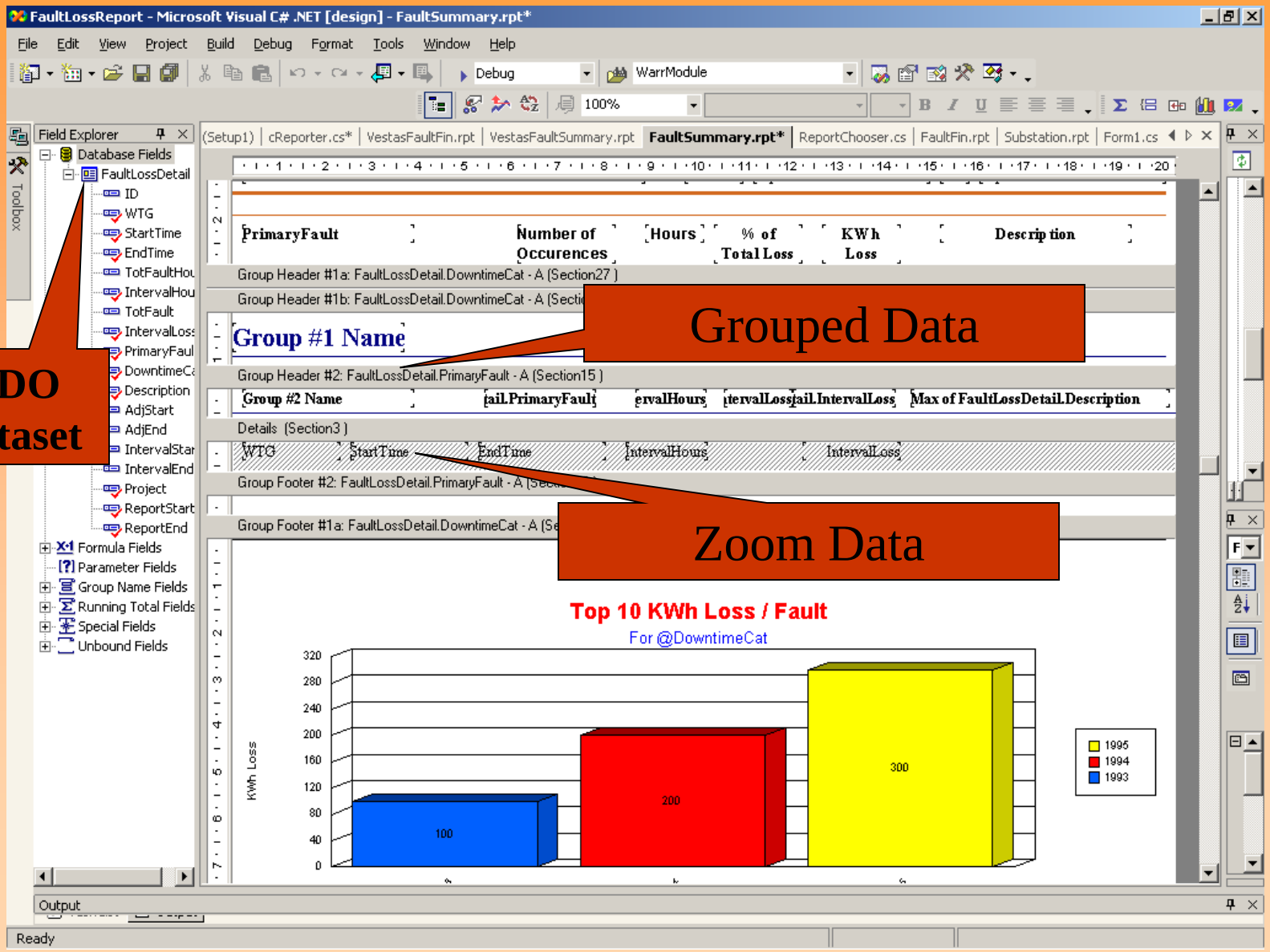
```
PISDK.PIModule GeneratorMod;
GeneratorMod = WarrModule.PIModules[“CN_I_01-01”].PIModules[“Generator”];

PISDK.PIAlias pa;
pa = GeneratorMod.PIAliases[“RPM”];

PISDK.PIPoint pt;
pt = mySDK.GetPoint( @"\\" + @strServer + @"\" + @pa.DataSource.Name);

DataEnd = pt.Data.ArcValue(dtEnd,PISDK.RetrievalTypeConstants.rtInterpolated,null);
```

# Visual Studio 2003: Crystal Reports



# Visual Studio 2003: How To

Call Crystal Report Form

```
public dsFD ds4FaultReport
{
    set
    {
        ds = value;
        // Just set the report source, everything else is done from the
        //report chooser form.
        crReportDocument.SetDataSource(ds);
        //Set the viewer to the report object to be previewed.
        crystalReportViewer1.ReportSource = crReportDocument;
    }
}
```

# Lessons Learned

- **Visual Studio, Dots, & the Module Database:**
  - Traversing the module database can be slow if not done properly.
  - Slow Method:

```
for (int i = 1; i <= modProj.PIModules.Count; i++)
{
        lbWTGs.Items.Add(modProj.PIModules[i].Name);
}
```

  - Faster Method:

```
PISDK.PIModules mods = modProj.PIModules;
foreach(PISDK.PIModule mod in mods)
{
        lbWTGs.Items.Add(mod.Name);
}
```

# Questions?