Authors' final copy of paper published in Computers & Chemical Engineering, 31, 2006, pp 86-99.

Computers & Chemical Engineering, *Accepted for publication*, May 2006

Using Process Topology in Plant-Wide Control Loop Performance Assessment

S.Y Yim*, H.G. Ananthakumar*, L. Benabbas*, A. Horch+, R. Drath+ and N.F. Thornhill*ᵡ

**Department of Electronic and Electrical Engineering, University College London, Torrington Place, London WC1E 7JE*

+ABB Corporate Research Centre, Ladenburg, Germany

Abstract

This contribution describes how disturbances in a control system can be isolated and diagnosed automatically based on plant topology. In order to demonstrate this, a prototype software has been designed and implemented which, when given an electronic process schematic of a plant and results from a data-driven analysis, allows the user to pose queries about the plant and to find root causes of plant-wide disturbances. This hybrid system puts together two new technologies: the plant topology information written in XML according to the Computer Aided Engineering Exchange (CAEX) schema and the results of a signal analysis tool called Plant-Wide Disturbance Analysis (PDA). The isolation and diagnosis of the root causes of plant-wide disturbances is enhanced when process connectivity is considered alongside the results of data-driven analysis.

Keywords: Fault detection and diagnosis, plantwide oscillation; plant topology; process monitoring; root cause; XML.

1. Introduction

Methods for data-driven, signal-based analysis have been developed in the past few years for finding root causes of plant-wide disturbances using measurements from routine process operations [Ruel and Gerry, 1998; Xia and Howell, 2003; Thornhill, 2005]. Several authors have observed, however, that data-driven analysis is enhanced if a qualitative model of the process is used as well to capture the fundamental causal relationships of a process in a non-numerical way [Chiang and Braatz , 2003; Lee *et.al.*, 2003]. Qualitative process information is implicitly used in diagnosis when an engineer considers the results from a data-driven analysis and an exciting possibility is to automate the use of such information. An industrial example in Thornhill *et.al.*, (2003) discussed the reasoning used by process control engineers to verify a signal-based diagnosis and to resolve ambiguities. The qualitative information used in the analysis was the connectivity between items of plant equipment displayed in the process schematic and the locations of indicators and control loops. The challenge now is to capture that information in electronic form and to manipulate it to draw conclusions.

Object-oriented representations of processes are becoming widely available using computer aided engineering tools such as ComosPT from Innotec, and Intools or SmartPlant P&ID from Intergraph. The plant topology as described in process diagrams can now be exported into an vendor independent and XML-based data format, giving a portable text file that describes all relevant equipments, their properties and the connections between them [Fedai and Drath, 2005]. The Standard is described in DIN V 44366 (2004) and IEC/PAS 62424 (2005) which is called Computer Aided Engineering Exchange (CAEX) and which specifies an XML schema. ISO-15926-7 is a similar standard.

A prototype tool called CAEX Plant Analyser that links a CAEX description with a report from a signal-based plant disturbance analysis tool (PDA) is reported in this article. The features are:

- Capture of process topology using CAEX;
- Parsing and manipulation of the description;
- Linkage of plant description and results from data-driven analysis;
- Generation of root cause hypotheses;
- Logical tools to give root cause diagnosis and process insights.

The CAEX file describes items of equipment in the plant such as tanks, pipes, valves and instruments and how they are linked together physically and/or through electronic control signals (the plant topology). The term *topology* is used here in its meaning as the physical structure of a network. The PDA report file gives information about the plant disturbances, for instance the period, intensity and regularity of an oscillation, the measurement points where it was detected and any non-linearity detected in the time trends.

A reasoning engine finds physical paths and control paths in the plant and the connections between items of equipment, and determines root causes for plant-wide disturbances. It can also verify that there is a feasible propagation path between a candidate root cause and the other locations in the plant where secondary disturbances have been detected.

ᵡ Corresponding author:
e-mail: n.thornhill@ee.ucl.ac.uk
Tel: +44 20 7679 3983

The system is a working prototype that combines a qualitative process model with data-driven analysis in a new way, exploiting the opportunities of CAEX for representation of the process connectivity. It integrates a parser reading a CAEX XML file, a file of numerical results from signal-based analysis and a reasoning engine within a graphical user interface. That there is a need for progress in this area is clear e.g. from Maurya *et. al.*, (2004) who commented that there are hardly any research articles dealing with the qualitative analysis of large process flowsheets. This paper is therefore amongst the first contributions to research in this field and it shows the CAEX standard for representation of process information in XML can make a contribution to plant-wide process monitoring and diagnosis.

Section 2 of the paper describes the background and places the work of this paper in context. Section 3 describes a parser which interprets the XML and PDA files, and the reasoning engine which has been written in Prolog. A case study is presented in Section 4, which also illustrates a graphical user interface which allows the user to pose queries. The paper ends with a critical evaluation of the contributions of the work and a conclusion.

2. Background and context

The CAEX Plant Analyser is a hybrid system because it combines a qualitative process model with analysis of quantitative process history data. This section places qualitative modelling in the context of other approaches such as signed directed graphs (SDGs) and also discusses quantitative process data analysis for plant-wide control loop performance assessment and diagnosis.

Quantitative and qualitative process models: The review series by Venkatasubramanian *et. al.*, (2003*a*, 2003*b*, 2003*c*) discussed methods for detection, isolation and diagnosis of faults in chemical processes, classifying the available methods into quantitative and qualitative model-based methods and quantitative and qualitative process history based methods.

Quantitative model based methods for fault detection and isolation (FDI) seek for inconsistencies between observed behaviour and the behaviour predicted by a detailed model of a process. FDI based on first principles models is a challenge in the process industries because accurate, calibrated and validated models are expensive to create and maintain

The Signed Digraph (SDG) is one of the main ways of representing causal qualitative knowledge. Maurya *et. al.*, [2003*a*, 2003*b*] gave a comprehensive review of graph-based approaches for safety analysis and fault diagnosis of chemical process systems and showed how to develop graph models systematically from a system of differential-algebraic equations. Their utility in qualitative analysis of process flowsheets was also demonstrated. If the SDG is derived from an equation-based model of the process, though, then the limitations related to cost and maintenance would apply.

Quantitative process data analysis: Quantitative process history based methods use measurements from the process. The methods of interest in this article are those which detect and characterize plant-wide disturbances in a process control system.

Signal analysis for detection of oscillation has been reported [Hägglund, 1995; Forsman and Stattin, 1999; Miao and Seborg, 1999; Salsbury and Singhal, 2005; Thornhill and Hägglund, 1997]. For plant-wide detection a characterization and clustering step is needed in addition to oscillation detection giving a report stating which measurement is disturbed by each oscillation [Thornhill *et.al.*, 2003*b*]. Persistent non-oscillatory disturbances are characterized by their spectra which may have broad-band features or multiple spectral peaks. Plant-wide detection involves a suitable distance measure by which to detect similarity and clusters of measurements with similar spectra [Thornhill *et.al.*, 2002; Xia and Howell, 2005; Tangirala *et.al.*, 2005].

Signal analysis is also used for diagnosis of both non-linear and linear root causes. Examples of non-linear sources include control valves with excessive static friction, on-off and split-range control, process non-linearities leading to limit cycles, and hydrodynamic instabilities such as slugging flows. Early studies used the presence of prominent harmonics as an indicator of non-linearity [Thornhill and Hägglund, 1997; Ruel and Gerry, 1998], while recent methods include analysis of the bispectrum [Choudhury *et.al.*, 2004] and non-linear time series analysis. A non-linear time series means a time trend which is the output of a non-linear system. Root cause diagnosis based on non-linearity has been reported [Thornhill *et.al.*, 2003*a*; Thornhill, 2005; Zang and Howell, 2004; Zang and Howell, 2005] on the assumption that the measurement whose time trend has the highest non-linearity is closest to the root cause. Tests based on routine operating data specifically for the detection of sticking valves have been recently reviewed and compared by Rossi and Scali, 2005.

Linear root causes of plant-wide disturbances include poor loop tuning, controller interactions and recycle problems involving coordinated flows of mass or energy. Detection and diagnosis of some of these problems using signal-based analysis is starting to be reported [Xia and Howell, 2003; Zang and Howell, 2003; Bauer *et.al.*, 2004] but it is not yet as advanced as detection and diagnosis of non-linear root causes.

Qualitative and hybrid methods: Qualitative methods use process insights from engineers and operators derived from past experiences. An early and successful expert systems project in the UK was published by Blue Circle Industries (1990) while Harris *et.al.*, (1996) used an expert system to enhance the diagnosis of a control loop performance assessment system in a newsprint mill. Norvilas *et. al.* (2000) and Tatara and Cinar (2002) have also successfully combined multivariate statistical data analysis with expert systems for process fault diagnosis. Cowan (2001) reviewed expert system applications and concluded that systems with fixed goals are more successful than those which have to balance conflicting goals. The relevance of these comments is that the reasoning engine in CAEX Plant Analyser is rule based and could be regarded as a simple expert system. The threats identified by Cowan are minimised because the outcomes of the reasoning engine can be determined algorithmically from the facts parsed from the process schematic.

Signal analysis of a plant-wide disturbance can go some way towards finding the root cause of a disturbance. The results do not, however, take account of physical relationships and connections between the measurements. A knowledge of the process flowsheet enhances the diagnosis, in particular about which loops might disturb one another.

Stanfelj *et al.* (1993) provided a decision-making tree which included cross-correlation between a feed forward signal and the controlled variable of the loop under analysis. Likewise, Owen *et.al*., (1998) showed an application of control-loop performance monitoring in paper manufacturing which accounted for upset conditions of the whole mill and interactions between control loops. Chiang and Braatz (2003) made similar observations. Leung and Romagnoli (2002) integrated a multivariate statistical analysis method with cause and effect map of a process, which was set up manually, to help in the diagnosis of faults, while Lee *et.al.*, 2003 also combined SDGs with multivariate statistical analysis for enhanced diagnosis. It is clear that signal-based analysis is enhanced by the capture and integration of cause and effect information from a process schematic, a step which now can be automated by the use of an XML representation

3. The CAEX plant analyser

3.1 Overview

An overview of CAEX Plant Analyser is presented in Figure 1. One input is the CAEX input file which describes the items of equipment in the plant such as tanks, pipes, valves and instruments and how they are linked together physically and/or through electronic control signals. An example of a physical link (or path) is a pipe carrying a flow of mass or energy, while an example of a control link (or path) is a cable connecting a valve to a controller carrying an electronic signal.

The PDA input file contains information about the plant disturbances, for instance the period of oscillation, its intensity and regularity, the measurement points where each disturbance was detected and any non-linearity detected in the time trends.

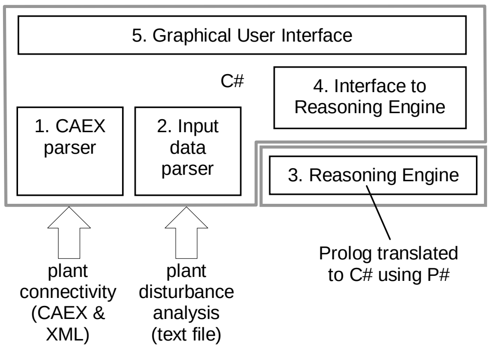

5. Graphical User Interface
C#
4. Interface to Reasoning Engine
1. CAEX parser
2. Input data parser
3. Reasoning Engine
plant connectivity (CAEX & XML)
plant disturbance analysis (text file)
Prolog translated to C# using P#

Figure 1. Overview of the CAEX Plant Analyser

<u>The system components</u>: The system architecture consists of five components each with a primary function and each designed using the object-oriented approach. The purpose of the parsers is to read and deconstruct the XML file containing the CAEX description and the PDA results text file. The XML parsing leads to lists of items of equipment and their connections from which the algorithms in the reasoning engine find physical and control paths in the plant and the connections between equipments. With additional results from signal-based analysis it also determines root causes for plant-wide disturbances. It can also check that there is a feasible propagation path between a candidate root cause and all the other locations in the plant where secondary disturbances have been detected. The user interface makes it easy to present such queries and to read the answers.

<u>Programming and integration</u>: Two programming technologies were used to implement the system. Prolog was used to implement the functions of the reasoning engine and C# ("C sharp") for all the remaining parts of the system. This design combines the advantages of the two different types of programming languages. The declarative programming language Prolog exploits the rule based nature of the connectivity information and the procedural and object-oriented features of C# allow an efficient parser and graphical application. The use of P# [Cook, 2004*a* and 2004*b*] to translate Prolog code to C# leads to an integrated system as a standard Windows application.

Despite the translation of Prolog code into C#, information is still represented in different ways due to differing data types which exist in the two languages. Therefore, an interface to the reasoning engine was needed to convert data that must flow between the components into the correct format. Each service provided by the reasoning engine implemented in Prolog could be accessed by a single call to a function provided by the interface.

<u>Distinctiveness of the approach</u>: The distinctiveness of the CAEX approach should be considered in the context of successful recent developments in SDGs. In Maurya

et. al., (2004), for instance, the SGD was derived from a mathematical model of the Tennessee Eastman benchmark simulation process. It modelled the patterns of deviations in the measurements when a fault from the library of known faults was present. The CAEX process representation has less predictive capability than an SDG because it is derived from a process schematic and not from a mathematical model. It gives binary (yes/no) answers to queries about presence or not of physical links and control paths, but not the signs of deviations from a nominal operating point. On the other hand, it is well suited to operate with data-driven signatures such as signal non-linearity, oscillation period or spectral grouping. When such signatures are used to characterise a plant-wide disturbance the outcome is a list of measuring points where each plant-wide disturbance has been detected. The binary nature of the process model then works very well to verify or falsify hypotheses about the disturbances and thus to infer results such as root causes and propagation paths.

3.2 Core technology

A number of technologies, development tools and platforms were used for the work. A brief description of each is now given.

Extensible Markup Language and CAEX: XML is an open standard developed by the XML Core Working Group which forms part of the World Wide Web consortium (Quin, 2005). XML uses plain text to represent structured data and uses tags to mark up the information. The beginning of a section is marked with an opening tag and the end of a section is marked with an end tag. For instance, XML defines a section of a document called **`Plant`** as shown below:

```
        <Plant PlantName="Plant1">
        ...
        </Plant>
```

The tag **`</Plant>`** marks the end of this part of the document and hence the end of the description of the plant. Tags can also contain attributes, in the example the attribute is **`PlantName`** whose value is **`Plant1`**.

The structure and possible content of an XML document is defined in an XML Schema which specifies which tags are allowed and what attributes they can have. A valid XML document must conform to the schema and the schema for representing an industrial plant is defined in Computer Aided Engineering Exchange (CAEX). CAEX is a vendor independent data exchange format used to describe plant information including the describing of a hierarchical and interconnected plant topology. It was developed jointly by a consortium of companies among them ABB. The relevant standard is IEC/PAS 62424 (2005).

Comos PT and the CAEX Exporter: Comos PT (Innotec GmbH, Schwelm, Germany) is a Computer-Aided Design suite that includes object-oriented process diagrams, among many other features. It has a library of elements present in industrial and chemical plants such as pipes, condensers and columns from which to build up the plant topology. The CAEX exporter application creates an XML file compatible with the CAEX XML schema from Comos PT data.

The user has to make decisions about the level of detail to be included and the naming conventions. In the application presented here a top level process schematic such as the one shown in Figure 2 contains enough detail (the heavy highlighting in Figure 2 will be discussed later). Some planned future rules in the reasoning engine will distinguish between process and utilities streams to enable the analysis of root causes of plant-wide disturbances propagating through utilities.

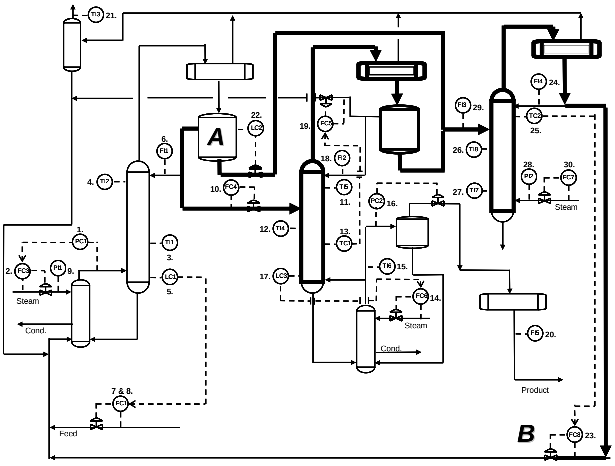


Figure 2. Process schematic (courtesy of J.W. Cox, Eastman Chemical Company).

Plant Disturbance Analyser (PDA): PDA has brought together published data-driven plant-wide disturbance analysis and diagnosis methods in one application within ABB's industrial product portfolio [Horch *et.al.*, 2005]. Its relevance to this article is that it produces the report giving the measurement locations where plant-wide disturbances have been detected, a characterization of the disturbance (e.g. the oscillation period, the signal non-linearity) and an ordering of the measurements in order of the amount of non-linearity.

Microsoft .Net: Microsoft .Net ("dot net") allows applications to integrate and communicate easily. It consists of the Microsoft .Net Framework, development tools to develop software for .Net, and client and server software. The .Net Framework is a run-time environment which allows programs built using different programming languages and running on different supported platforms to exchange data and work together through the use of platform-independent technologies such as XML. The framework hides the heterogeneity of the different environments and provides class libraries and a uniform environment.

The C# Programming Language: The whole of the CAEX Plant Analyser Application except for the reasoning engine is written in the C# programming language. C# was developed by Microsoft and contains similarities with Java and C++ and uses object-oriented features. It is supported by .Net and includes a number of classes that aid working with XML and the creation of Graphical User Interfaces.

Prolog: Prolog (PROgramming LOGic) is a declarative programming language which uses rules called predicates and facts to determine whether a query is true or false. It develops knowledge from the predicates and facts. For example, given that pipe1 is connected to valve1, and valve1 is connected to pipe2, and that a *path* is a list of connections, it is clear that there exists a path between pipe1 and pipe2. Prolog is able to determine that this is logically the case by applying its rules. The Prolog environment was P# which was developed by J.J. Cook in the Laboratory for Foundations of Computer Science at the University of Edinburgh. It includes a subset of the Prolog programming language as a native implementation language for the .NET platform and interoperation is achieved by means of C# objects created from Prolog.

3.3 The CAEX and input data parsers

CAEX file structure: A CAEX file is organised in a tree structure. A CAEX file contains four top level sections, **SystemHierarchy**, **InterfaceClassLib**, **RoleClassLib** and **SystemUnitClassLib**. The main relevance to the work in this article is the **SystemHierarchy** section, the structure of which is given in Figure 3.

Each **SystemHierarchyElement** section has an attribute called **SystemUnitInstanceName** and represents a separate **SystemUnitInstance**. The **SystemUnitInstanceName** gives a description of the topology of a section of a plant.

Within **SystemHierarchyElement** is a section called **SingletonClassDescription**. Inside this are the **InternalElements** and **InternalLinks**. Each **InternalElement** section represents a particular unit in the plant therefore every tank, pipe, condenser and so on has an entry in this section of the document. The description of plant items in CAEX is recursive and an **InternalElement** may contain further **InternalElement** sections. An **InternalLink** represents a link between two elements in the plant, for example, a pipe that is joined to the input of a valve will have an **InternalLink** entry which specifies this information. The list of **InternalLinks** describes the topology of the plant.

Parsing: Parsing involves reading a text document, identifying key words and symbols and converting the information into a data structure which can be operated upon and manipulated. The parser in the CAEX Plant Analyser application has two modules (1 and 2 in Figure 1). The CAEX Parser reads the CAEX file and produces the Plant Object Model which is an object-oriented representation of the plant topology. The Data File Parser reads the PDA report file and organizes the information within it.

CAEX Parser: The CAEX parser takes as input an XML document conforming to the CAEX standard. It also performs the functions listed in Table 1 so that it can provide the graphical user interface and the reasoning engine with information about the plant.

The parsing code is contained within a function which takes as its input the name of the CAEX file, parses the file and returns the number of internal links found. It traverses the tree structure of the CAEX file looking for **InternalLink** tags. When such a tag is encountered, the link name and the two elements at either end of the link are read and stored. The algorithm includes a check to determine if the element has been encountered before, and if it is new then a new Element object is added to the list of elements encountered so far. An **InternalLink** object of the same name is created containing references to the two Element objects which are present in the link and added to the list of **InternalLinks** present in the topology.

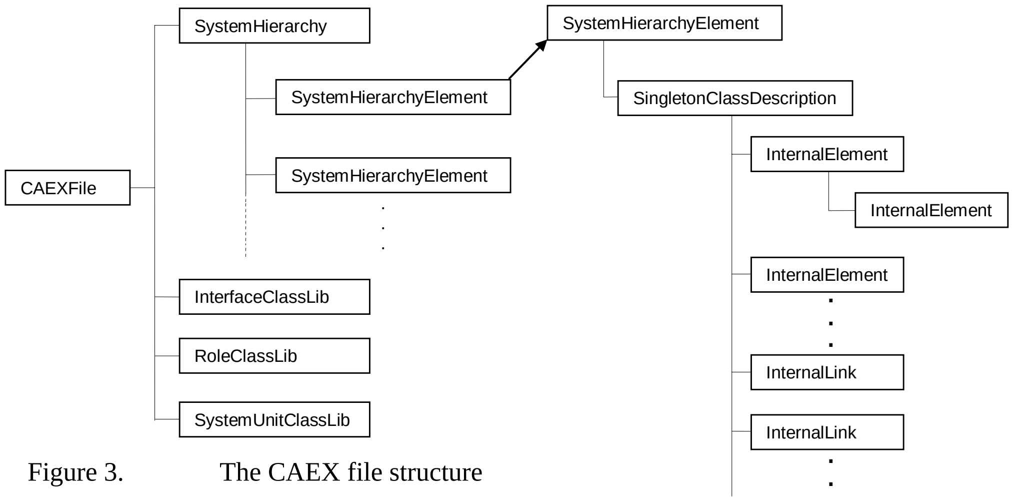


Figure 3. The CAEX file structure

<u>Input data parser</u>: The input data file is the PDA report containing the results from plant-wide disturbance analysis of measurements from the indicators and controllers in the operating plant. The information in the file includes:

- The period of oscillation [Thornhill *et.al.*, 2003b];
- The standard deviation in the period of oscillation;
- The signal power present in the oscillation;
- The non-linearity index [Thornhill, 2005];
- The plant-wide disturbance or disturbances which affect the element.

The function of the input data parser is to read this data file and manipulate it to enable the reasoning engine and User Interface to access the required data. One of its functions is to convert the CAEX name of an element into a legal Prolog name by removing characters and features that are invalid in Prolog.

Table 1. Functions provided by the CAEX parser

Functions provided by the CAEX Parser
Return the list of elements in the plant.
Return the list of the links in the plant.
Return the name of the element at the input and output of a specified link.
Return the number of links present in the topology.
Return the number of elements present in the topology.
Search the topology to find out if a specific element exists in it.
Search the topology to find out if a specific link exists in it.

3.4 The Prolog reasoning engine

The reasoning engine uses the information that has been parsed from the CAEX representation of a plant schematic to give responses to queries. The methods in the prototype include:

- Path finding (physical path, control path, directly connected elements);
- Analysis of control loops and indicators (non-linear controllers and indicators, disturbance-rejecting controller, proxy measurement);
- Root cause analysis (root cause diagnosis, faulty elements).

This section describes how the reasoning is achieved.

<u>Prolog basics</u>: Facts in Prolog define the characteristics of specific objects. The facts in the CAEX Plant Analyser are parsed from the CAEX description and are sent dynamically to the reasoning engine from the Parser. An example of a fact is:

```
link(pfb1lc2,pfb1fi3).  % pfb1lc2 is
                        %linked to pfb1fi3
```

Facts can be queried by a Prolog program or by direct interaction in a Prolog environment. The queries below illustrate that links are directional, thus allowing the direction of flow to be encoded:

```
?- link(pfb1lc2,pfb1fi3)
yes
```

```
?- link(pfb1fi3,pfb1lc2)
no
```

Lists are data structures such as:

```
[tank 1, tank 2, tank 3]
```

Rules are statements about relationships between objects which use the syntax **:-** (*if*), a comma **,** (*and*) and a semicolon **;** (*or*). The next example is a predicate called **`physicaltravel`** which uses a rule to test if there exists a link between X and B that is not via signal lines. If the answer is yes then it adds input argument B to list Path. Called recursively, this line of code builds up a physical path in the list Path.

```
physicaltravel(X,B,Path,[B|Path]) :-
       link(X,B),
       not(signalline(X)),
       not(signalline(B)).
```

<u>Path finding</u>: Path finding provides information to the user about physical paths and control paths through the plant between specified start and end points. A physical path is via pipes and other physical elements such as columns. The control path is via signal lines, controllers and indicators. The program receives as inputs the specified start and end points and uses the list feature in Prolog to build lists of the intermediate elements and thereby to find path. There may be multiple paths and also no paths; both of these cases are handled.

The Prolog code for finding a physical path is equivalent to the procedural algorithm shown in Figure 4. In the case of multiple paths from any element the paths are each followed in turn. All the paths are returned and the numbers of elements in each path are compared by the C# calling function to find the shortest. The presence of a recycle can be detected searching for paths with the same element as the start and end. This example illustrates how Prolog reasons from rules given a set of facts. Control paths are determined in a similar manner with the constraint that the path is via signal lines.

<u>Finding control loops and indicators</u>: Basic queries such as **`controller(X)`** find controllers, indicators and signal lines. A predicate called **`controlloop`** determines other physical elements that are part of a loop. The rule is that there is a signal line coming into the element from a controller or there is a signal line that leaves the element that is linked to a controller.

<u>Indicator and controller status</u>: Queries about the operational status of indicators and controllers use the plant topology and the results from signal-based PDA analysis. Control loops with non-linearity present in the process variable (PV), output (OP) or set point (SP) are identified from the PDA results and so are controllers achieving disturbance rejection. The rule for disturbance rejection is that the OP is a member of a disturbed cluster while the PV is not. Physically this indicates variability being successfully diverted from the controlled variable to the manipulated variable.

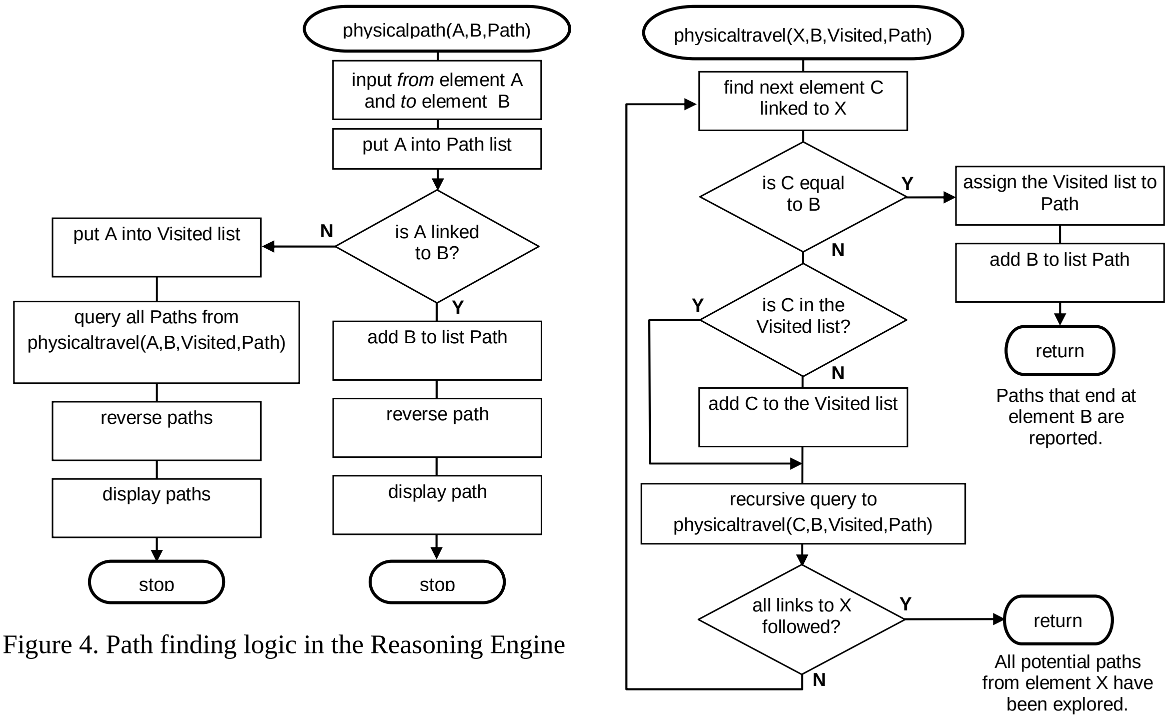

physicalpath(A,B,Path)
input *from* element A and *to* element B
put A into Path list
is A linked to B?
N
Y
put A into Visited list
query all Paths from physicaltravel(A,B,Visited,Path)
reverse paths
display paths
stop
add B to list Path
reverse path
display path
stop
physicaltravel(X,B,Visited,Path)
find next element C linked to X
is C equal to B
Y
N
assign the Visited list to Path
add B to list Path
return
Paths that end at element B are reported.
is C in the Visited list?
Y
N
add C to the Visited list
recursive query to physicaltravel(C,B,Visited,Path)
all links to X followed?
Y
N
return
All potential paths from element X have been explored.

Figure 4. Path finding logic in the Reasoning Engine

Other rules exploit the integration of the data-driven analysis and the plant topology to test for loops in manual and to detect ratio, feedforward and cascade configurations. For instance, the rule to detect a cascade configuration is that the OP of the master loop is equal to the SP of the slave loop.

Automated root cause diagnosis: The prototype CAEX plant analyser uses plant topology to automate the finding of root causes due to non-linear effects such as sticking valves. Its purpose is to use information about process layout and connectivity from the CAEX file to enhance the signal-based analysis. It resolves ambiguities and verifies that feasible propagation paths exist from the proposed root cause to other places in the plant where the disturbance has been detected.

Studies have shown that control loops whose signals have non-linearity are candidates for the root cause. Predicates for dealing with the non-linear case include: **possiblerootcausecontrollers** which finds controllers with non-linearity in both PV and OP, **possiblerootcauselements** which find the valve connected a possible root cause controller and **faultyelement** which finds plant elements connected to indicators where non-linearity has been detected. Faulty elements give a reference point to find which element is upstream and downstream. The key feature of the **rootcause** predicate is that it looks for evidence that non-linearity has spread.

The **checklink** predicate finds valves A such that A is connected (linked) to B and B is linked to C, subject to the constraints that valve A is a **possible-rootcauseelement** of disturbance group N, that B is a pipe and C is a pipe connected to a faulty element. Links are directional and hence the **checklink** predicate finds upstream valves. The **rootcause** predicate returns A as the root cause if the **checklink** predicate finds such a valve. It also returns A as the root cause if C is linked to D and D is linked to A, where C is a faulty element and D is any element. This takes care of cases such as level controls where the level is controlled on the outflow. If more than one root cause is returned then a further rule chooses the one that has paths to the others, i.e. the one furthest upstream.

3.5 Interface to the reasoning engine

This section explains the way in which the plant object model is passed to the reasoning engine. The reasoning engine was developed in Prolog and then translated to C#, hence original Prolog data-types are transformed into C# objects. The interface class includes the code which hides the heterogeneity of the data types and passes the input objects to the reasoning engine and returns the results in the correct format.

Calling Prolog predicates from C#: Prolog predicates can be called from C#. The first thing to be done is to add the calling assembly, this is done by the **sharp.AddAssembly** instruction. An example would be a call to a Prolog predicate called **createBasin**.[1]. After translation from Prolog to C# using P#, the predicate name becomes a C# method called **Create_Basin** which calls the predicate and finds the answer in Prolog.

Reading plant objects: The parser passes the plant connectivity information to Prolog. These details must

[1] A *basin* is a source or sink at the plant boundary.

be read by Prolog and then added to its database of facts in order to deal with the query the user may pose. Prolog adds facts to the reasoning engine using a predicate called **`assert`**. To add the facts that describe the plant topology to the Prolog database, C# cycles through the plant elements and makes a call containing the predicate **`assert`** to the reasoning engine.

Interface functions: The user can pose queries such as: physical path; control path; control loop; directly connected elements as described in Section 4. In Prolog, the physical path algorithm returns the path between two elements via pipes and other elements, and the control path returns the path via signal lines, controllers and indicators only. For these queries the element from which the path is required and the element at the end point of the path are specified.

A number of different paths may be found and all the results are packed into a data structure called an **`ArrayList`**. The **`ArrayList`** is further manipulated in order to put the shortest path into the first position of the **`ArrayList`**. This was done by C# manipulating the string produced by the reasoning engine. Finally, the paths are presented in the graphical user interface.

The principles of the interfaces for other queries are similar, though the type of answer may be different, for instance a query whether an element is in a control loop returns a yes/no answer.

The outputs of queries related to the analysis of root causes contains a list of elements along with their associated group of oscillation. The user of the application is interested in knowing the element that is the root cause of a fault and the oscillation group it belongs to (and hence which fault it is causing).

3.6 Graphical user interface and integration

The features of the graphical user interface (GUI) are implemented in the class **`CAEX Plant Analyser`**. The CAEX Plant Analyser class and GUI provide the framework which integrates all parts of the program. In addition to presenting a visual environment to the user, the interface also performs all the actions that make the program work at run-time, such as execution of the **`PlantParser`**. The functions of the GUI are shown in Table 2.

The user sees the main program window which has a menu bar and five tabs which display information and allow the user to enter queries. An example is presented in Section 4. The GUI also has Tooltips which tell the user what a particular button or region does. These are displayed when the mouse pointer is over the feature.

Integration: The system in C# is made up of a number of components. Each system component is termed a C# *project*, these contain the classes which implement the functions of the system. The collection of C# projects that make up the full system is called the *solution*.

The solution created for the system is made up of four C# projects, **`CAEX Plant Analyser`**, **`PlantRepresentation`**, **`PrologQuery`** and **`Specifics`**. The **`PlantRepresentation`** project contains everything required to read, understand and manipulate the relevant information in a CAEX file and PDA input data. **`PrologQuery`** contains the reasoning engine and interface. The **`Specifics`** project contains the translation of the naming conventions in the process flowsheet. The CAEX standard offers no semantics or meaning so it is not possible to tell what is a valve or a tank just from its name. A mapping is therefore needed between the names of objects from the CAEX representation and the names of objects from other sources such as from PDA. Table 3 shows the projects present in the overall C# solution and the functions of the classes within each project.

On compilation, three projects **`Specifics`**, **`PlantRepresentation`** and **`PrologQuery`** become libraries of functions called Dynamic Link Libraries (DLLs) in the Windows operating system. The CAEX Plant Analyser project compiles into an executable file. There is an additional file **`Psharp.dll`**, a pre-compiled library and part of P#, which contains C# definitions of classes and data types which are required for Prolog calls. The **`PrologQuery`** project makes use of the definitions contained in this library. The overall system is a Windows .Net application and can be started by double-clicking on the **`CAEX Plant Analyser.exe`** icon generated when the C# solution is compiled.

Table 2. Functions of the Graphical User Interface

Item	Function
File menu	Use the “Open CAEX file” item in this menu to choose and open a CAEX file. Use the “Open data file” item in this menu to choose and open a data file.
Display Plant Diagram	Allows the user to select a bitmap image to display a picture of the plant they are querying.
Internal links	Displays information about the internal links present in the opened CAEX file.
Elements	Displays information about the internal links elements present in the opened CAEX file.
Input data	Displays information about the data in the opened data file.
Perform queries	Performs queries on the plant topology such as find the shortest path between two elements in the topology.
Root cause	Displays details about working and faulty controllers and indicators and possible causes of faults in the plant.
Information area	Plant and data information is displayed in this area.
About	Displays the About Box.

Table 3. Functions of the C# projects and classes

C# Project	Classes	Function of Classes
Plant Representation	Element	Represents and allows manipulation of a plant element
	InternalLink	Represents and allows manipulation of an InternalLink
	PlantParser	Contains functions which parse a given CAEX file, extracting the elements and InternalLinks and creating the Plant Object Model
	InputData	Contains functions which parse a data file containing the results from Plant-Wide Disturbance Analysis
Specifics	Specifics	Implements the naming convention used in the CAEX file representing the plant. Identifies the element type (pipe, controller, etc.) when given the CAEX name.
Prolog Query	PrologQuery	Presents a query to the reasoning engine and returns the result.
CAEX Plant Analyser	MainForm	The main CAEX Plant Analyser GUI window.
	AboutBoxForm	The About Box window of the CAEX Plant Analyser application, appears when the user selects the “About” menu option.
	PlantDiagramForm	The PlantDiagram window, appears when the user chooses to display a Bitmap diagram of the plant.

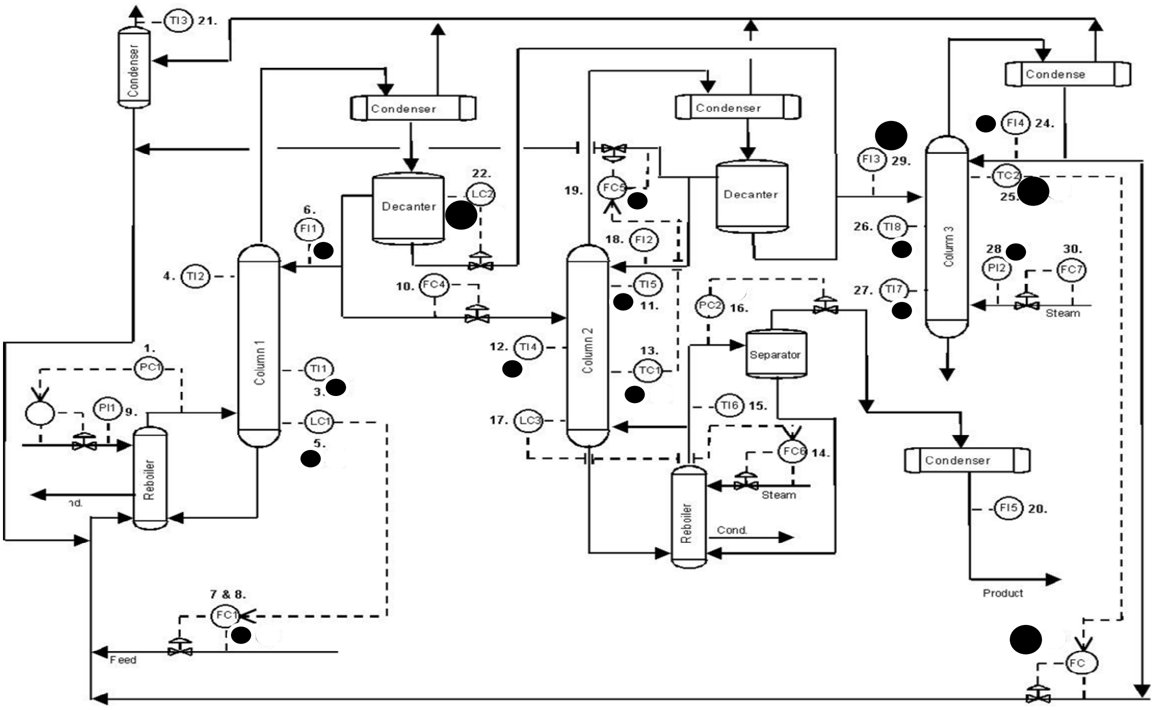


Figure 5. Process schematic for the case study. The black spots show the locations of a plant-wide disturbance and large spots indicate measurements whose time series showed non-linearity.

4. Example application

The application is from the Eastman Chemical Company and was originally presented in Thornhill *et.al.*, (2003*a*). That paper described signal-based analysis using measurements from the plant and also described the manual steps using process information that the process control engineers took to resolve ambiguities and check the findings. The work of this paper was motivated by that study, which called for an automated way of incorporating information about the process layout and connectivity.

Detection of plant-wide disturbances: Figure 5 shows the process schematic. The black spots in the schematic (placed by hand) indicate the locations where one of three plant-wide disturbances was discovered by signal analysis using the PDA tool. The larger spots show the locations of measurements whose time series were non-linear and therefore candidates for the root cause (see Thornhill, 2005). The PDA report sent to the CAEX Plant Analyzer gives the following information for each measurement point in the plant:

- The period, regularity and strength of any oscillation present;
- The plant-wide cluster to which the oscillation or disturbance belongs;
- The amount of non-linearity detected in the time trend.

In the previous paper (Thornhill *et.al.*, 2003*a*), the manual reasoning steps taken by the engineers towards deducing the root cause of a disturbance were:

- Distinguishing measurement points that are part of a control loop from those which are indicators. The logic behind this is that control loops can generate non-linearity and are thus candidate root causes but passive indicators cannot;
- Determining connections between measurement points in the disturbed cluster. The reasoning is that disturbances are likely to propagate in the direction of the process flow;
- Identifying a proxy (if one exists) for an unmeasured quantity of interest;
- For a candidate root cause, establishing that a feasibly propagation path exists to other places in the plant showing the same disturbance.

CAEX Plant Analyser aids the engineer by using the connectivity model of the process to determine which among the candidates is the most likely root cause, and by presenting evidence to support the conclusion. It therefore automates the manual steps itemized above and enables the engineer to become involved at a higher level. The remaining tasks are to physically confirm the diagnosis, for instance by means of a valve travel test, and to write the maintenance order.

Root cause analysis: Figure 6 shows the GUI. The Root Cause tab is the part of CAEX Plant Analyser where the process connectivity information is linked with the data-driven analysis. Examples of the queries it can answer are:

- Disturbance-rejecting controllers, where the rule in use is that the OP is disturbed while the PV is not disturbed;
- Non-linear controllers, i.e. with non-linearity in PV or OP or SP);
- Non-linear indicators (those with non-linearity in the PV);
- Possible root cause controllers (those with nonlinearity in PV and OP);
- The actual root cause selected from among the non-linear controllers using the **`rootcause`** predicate described in section 3.4.

The control valve of the control loop LC2 (Tag 22) was identified as the root cause of the plant-wide disturbance whose distribution is mapped in Figure 5. The automated reasoning steps are that (a) the root cause is associated with one of the measurement points where there is non-linearity, (b) the root cause cannot be an indicator, which rules out Tag 29 (FI3), (c) the **`rootcause`** predicate determined that non-linearity has spread downstream from LC2.

Confirming the root cause: The previous sub-section indicated how CAEX Plant Analyser used the signal analysis results from PDA and the process connectivity model to propose the LC2 control loop as the source of an oscillating disturbance. The control engineers can now test the hypothesis. One standard test is an OP-MV plot, where MV is the manipulated variable for the LC2 control loop and OP is the level controller output. In this case, however, MV is the flow through LC2 control valve and is not measured.

The proxy measurement function in CAEX Plant Analyser suggests the use of FI3 (Tag 29) as a substitute for the unmeasured MV because it is the next downstream flow measurement. Although the flow in FI3 is not equal to the flow through the LC2 control valve, the logic behind its use as a proxy measurement is that the FI3 is a disturbed measurement in the same cluster. Its waveform should therefore be related to the disturbance to the unmeasured flow even though the numerical values in engineering units are not the same.

Figure 7 shows the time trends of LC2.OP and FI3.PV and the plot of one versus the other (the OP-MV plot). For a healthy valve the OP-MV plot should be a straight line at 45^o, however in this case it shows the classic feature of a valve with a deadband.

Further interventionist tests were conducted as described in Thornhill *et. a*l, 2003(a) which confirmed the diagnosis. At that time, the reasoning from plant topology was done manually, but now PDA and CAEX Plant Analyser can achieve the same result automatically. They direct attention to the right valve so that the amount of physical testing is minimized.

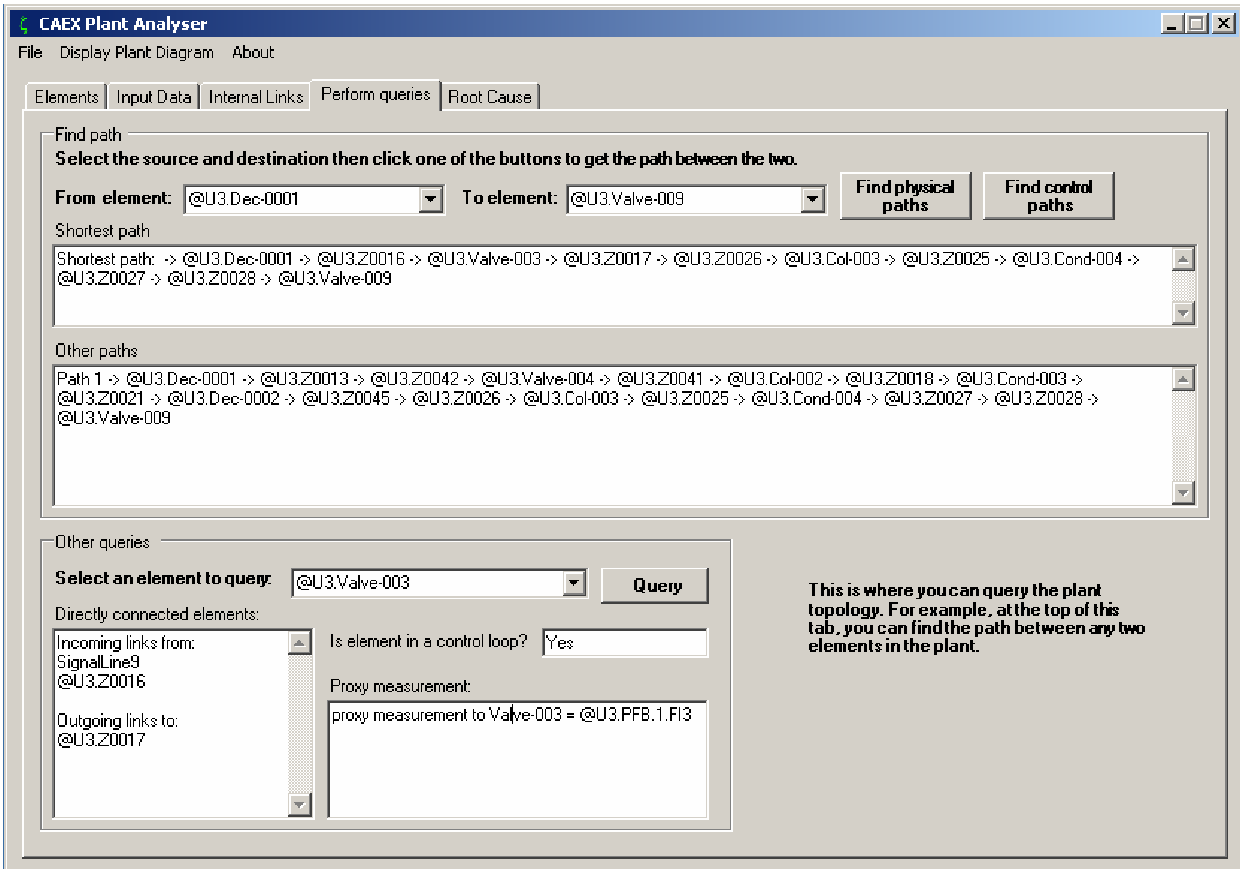
Propagation paths: A function of CAEX Plant Analyser is to help the user with understanding propagation of a disturbance from its root cause to all the other places in the plant where secondary upsets were detected. The **`Perform Queries`** tab (Figure 6) provides functions which find the physical and control paths, as described in Section 3.4.

In Figure 6, the upper half of the window shows physical paths or control paths between two selected elements. The output is in the form of a list such as is shown in the panel labelled Shortest path. The query requested the path in Figure 2 from decanter 1 marked as ***A*** (which is level-controlled by LC2) to the valve in the bottom right corner of Figure 2 marked ***B*** which controls the liquid recycle. For the purposes of visualization, the two physical paths found in the above example are shown in Figure 2. Between them, these physical paths link ***A*** to ***B*** via both columns 2 and 3. The disturbances to columns 2 and 3 and the recycle that were detected by PDA are thus consistent with the hypothesis of LC2 as the root cause.

The current prototype does not mark up the schematic so Figure 2 was produced manually. It is a target for the future to provide an animated schematic, but the best way to do this is not yet clear. The main issue is the extent to which an animated schematic would be integrated into displays already provided in the vendor's control system platform. This commercial decision is needed first because the extent of integration will guide the technical solution.

A further query asking for the paths both starting and ending at ***A*** (decanter 1) returned paths which highlighted all three of the recycles in the plant. These paths involve Column 1 and therefore illustrate a feasible propagation path from the root cause in LC2 to the disturbances detected by PDA in Column 1.

The lower half of the window in Figure 6 allows queries on individual elements in the topology. The program poses queries to the reasoning engine. It finds out which elements are directly connected to the selected one and displays the results. The reasoning engine also determines if the element is in a control loop. It will also find a proxy measurement, if one exists.



Select the source and destination then click one of the buttons to get the path between the two.

From element: @U3.Dec-0001 To element: @U3.Valve-009

Shortest path

Shortest path: -> @U3.Dec-0001 -> @U3.Z0016 -> @U3.Valve-003 -> @U3.Z0017 -> @U3.Z0026 -> @U3.Col-003 -> @U3.Z0025 -> @U3.Cond-004 -> @U3.Z0027 -> @U3.Z0028 -> @U3.Valve-009

Other paths

Path 1 -> @U3.Dec-0001 -> @U3.Z0013 -> @U3.Z0042 -> @U3.Valve-004 -> @U3.Z0041 -> @U3.Col-002 -> @U3.Z0018 -> @U3.Cond-003 -> @U3.Z0021 -> @U3.Dec-0002 -> @U3.Z0045 -> @U3.Z0026 -> @U3.Col-003 -> @U3.Z0025 -> @U3.Cond-004 -> @U3.Z0027 -> @U3.Z0028 -> @U3.Valve-009

Select an element to query: @U3.Valve-003

Directly connected elements:

Incoming links from:
SignalLine9
@U3.Z0016

Outgoing links to:
@U3.Z0017

Is element in a control loop? Yes

Proxy measurement:

proxy measurement to Valve-003 = @U3.PFB.1.FI3

This is where you can query the plant topology. For example, at the top of this tab, you can find the path between any two elements in the plant.

Figure 6. The Perform queries tab in the GUI.

Plant Object Model and PDA report: The **`Elements`** and **`Internal Links`** tabs of the GUI present the plant object model. The **`Elements`** tab shows a list of all the elements found in the topology while the **`Internal Links`** tab shows a list of all the internal links and gives the elements at the input and output of the link.

The **`Input Data`** tab of the application is populated with the table of data read from the PDA report. For each measurement point, it shows the oscillation period, regularity and power, the plant-wide cluster number and the results from non-linearity analysis.

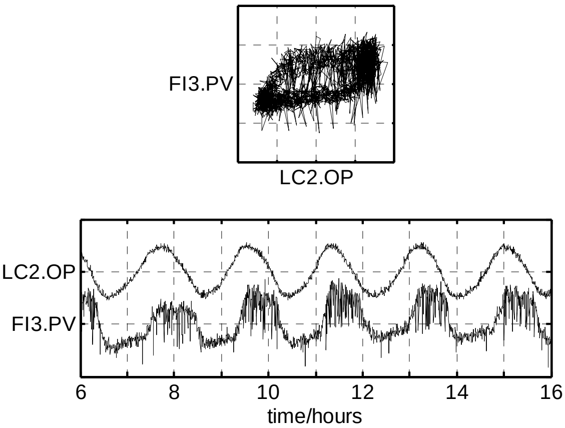


Figure 7. Plots of the LC2 control valve output (LC2.OP) and flow through the valve (FI3.PV).

5. Critical evaluation and conclusions

Requirements for a plant-wide control loop performance analysis system have been discussed by Paulonis and Cox (2003) and Desborough and Miller (2002), and more general criteria were laid down in Venkatasubramanian *et.al*., (2003*a*). Table 4 lists the relevant specifications. The prototype system has hit several of these targets.

Table 4. Statements of requirements for plant-wide control loop performance analysis

Authors	Requirements	met
Desborough and Miller (2002)	Facility-wide approaches including behaviour clustering;	✓
	Automated model-free causal analysis;	✓
	Incorporation of process knowledge such as the role of each controller.	
Paulonis and Cox (2003)	Detection of the presence of one or more periodic oscillations;	✓
	Detection of non-periodic disturbances and plant upsets;	✓
	Determination of the locations of the various oscillations/disturbances in the plant and their most likely root causes.	✓
Venkatasubramanian, Rengaswamy, Yin and Kavuri (2003*a*)	Rapid detection and diagnosis;	✓
	Ability to isolate as well as detect faults;	✓
	Ability to identify new and unseen faults;	✓
	Ability to diagnose multiple faults;	✓
	Ability to explain the reasons for decisions;	
	Minimal modelling requirement.	✓

Contributions: By itself, the signal-based analysis used in PDA reports clusters of multiple periodic oscillations and non-periodic disturbances. PDA in combination with CAEX Plant Analyser can offer hypotheses about the location and nature of the root causes, and the GUI makes it very easy to test the hypotheses for instance by presenting the propagation path from the root cause to all measurement points where the disturbance was detected by PDA. The CAEX Plant Analyser has incorporated process knowledge by linking the plant topology and a reasoning engine with the results from plant disturbance analysis. It gives enhanced automation to the process of disturbance detection and analysis by means of rule-base reasoning which suggests candidate root causes, establishes the propagation path and other process insights such as proxy measurements for quantities that are not monitored.

CAEX Plant Analyser provides automated causal analysis that is *almost* model-free. The main reason for preferring model-free analysis is related to the cost of producing a model and the difficulty of maintaining it. A plant topology description from a process schematic is a type of model. It is, however, easy to produce from a CAD tool. The maintenance of accurate drawings is a considerably easier and more routine task than the maintenance of a mathematical model or SDG. Modern integrated computer aided engineering tools mean that changes made to one document propagate automatically to other related documents including an updated CAEX file. Another motivation for model-free analysis the ability to detect and diagnose previously unseen faults. The CAEX plant analyser can do this because the signal-based analysis can characterize new signals and the CAEX Plant Analyser can work out the location of the root cause.

Limitations: Limitations of CAEX Plant Analyser in identifying the correct root cause stem from two sources. One is that the rule base is incomplete because the science of root cause diagnosis is not yet complete. If the root cause is a non-linear effect then the reasoning using signal non-linearity as described here is effective. The survey by Desborough and Miller (2002), and others, have shown that the majority of root causes are of these types. There remain, however, other causes of plant-wide disturbances such as controller interaction and structural effects such as the dynamics caused by recycles. These do not have non-linear signatures and enhanced signal analysis in PDA and rules to operate on the PDA results are needed for these cases.

Another limitation is that, even when the root cause is a non-linear effect, there may be no measurement available at the exact source of the root cause. In fact, this was the case in the example application in Section 4 and was addressed in that case by locating a proxy measurement for the unmeasured flow through the valve. The function of CAEX Plant Analyser is to help engineers form and test hypotheses. If no proxy is available then an outcome of a session with CAEX Plant Analyser might to collect a new data set for analysis with more measurements included from the region close to the suspected root cause.

Further targets are the annotation of a process schematic for visualization of disturbances and paths, and the ability to explain decisions. Explanations could be implemented through the Tooltips in the GUI. Progress with these aspects depends on commercialization because they will have to be integrated with the vendors existing systems and house style.

Conclusions: The CAEX Plant Analyser has given a new way forward to allow a user to pose queries about the plant and to gain insights into the root causes of plant-wide disturbances. To do this, it integrates an electronic description of the plant topology and results from signal-based analysis. Results produced from a working Windows prototype have been presented showing that the concept satisfies many of the requirements previously called for in the literature.

6. Acknowledgements

The authors would like to thank John Cox and Michael Paulonis of the Eastman Chemical Company for making the case study available. Nina Thornhill gratefully acknowledges the support of the Royal Academy of Engineering (Global Research Award) and the UCL team thanks ABB Corporate Research for their sponsorship of the research.

7. References

Bauer, M. (2005). *Data-Driven Methods for Process Analysis*, PhD Thesis, University of London.

Bauer, M., Thornhill, N.F., and Meaburn, A. (2004). Specifying the directionality of fault propagation paths using transfer entropy. *DYCOPS* 7 *conference*, Boston, July 5-7, 2004

Blue Circle Industries, plc. (1990). *Real Time Process Control: Improved Efficiency. Expert System Opportunities, Case Study 1*, HMSO, London.

Chiang, L.H., and Braatz, R.D. (2003). Process monitoring using causal map and multivariate statistics: fault detection and identification. *Chemometrics and Intelligent Laboratory Systems*, 65, 159-178.

Choudhury, M.A.A.S., Shah, S.L., and Thornhill, N.F. (2004). Diagnosis of poor control loop performance using higher order statistics. *Automatica*. 40, 1719–1728.

Cook, J.J. (2004*a*). P#: A concurrent Prolog for the .NET Framework, *Software: Practice and Experience*, 34(9):815-845.

Cook, J.J. (2004*b*). *Language Interoperability and Logic Programming Languages*, PhD Thesis, University of Edinburgh.

Cowan, R. (2001). Expert systems: aspects of and limitations to the codifiability of knowledge, *Research Policy*, 30, 1355-1372.

Desborough, L., and Miller, R. (2002). Increasing customer value of industrial control performance monitoring – Honeywell's experience, in *AIChE Symposium Series* No 326, 98, 153-186.

DIN V 44366. (2004). *Specification for Representation of process control engineering requests in P&I Diagrams and for data exchange between P&ID tools and PCE-CAE tools*, Beuth-Verlag, 2004.

Fedai, M., and Drath, R. (2005). CAEX – A neutral data exchange format for engineering data, *ATP International Automation Technology* 01/2005, 3, 43-51.

Forsman, K. and Stattin, A. (1999). A new criterion for detecting oscillations in control loops. *European Control Conference*, Karlsruhe, Germany.

Hägglund, T. (1995). A control-loop performance monitor. *Control Engineering Practice*. **3**, 1543-1551.

Harris, T.J., Seppala, C.T., Jofriet, P.J., and Surgenor, B.W. (1996). Plant-wide feedback control performance assessment using an expert-system framework, *Control Engineering Practice*, 4, 1297-1303.

Horch, A., Hegre, V., Hilmen K., Melbø, H., Benabbas, L., Pistikopoulos, E.N., Thornhill, N.F, and Bonavita, N. (2005). Root Cause - Computer-aided plant auditing made possible by successful university cooperation, *ABB Review* 2/2005, 44-48.

IEC/PAS 62424 (2005-6) Ed.1.0. (2005). *Representation of process control engineering requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*, VDE-Verlag, 2005.

Lee, G.B, Song, S.O.,. and Yoon, E.S. (2003). Multiple-fault diagnosis based on system decomposition and dynamic PLS, *Industrial & Engineering Chemistry Research*, 42, 6145-6154.

Leung, D., and Romagnoli, J. (2002). An integrated mechanism for multivariate knowledge-based fault diagnosis, *Journal of Process Control*, 12, 15-26.

Maurya, M.R., Rengaswamy, R., and Venkatasubramanian, V. (2003*a*). A systematic framework for the development and analysis of signed digraphs for chemical processes. 1. Algorithms and analysis, *Industrial and Engineering Chemistry Research*, 42, 4789-4810.

Maurya, M.R., Rengaswamy, R., and Venkatasubramanian, V. (2003*b*). A systematic framework for the development and analysis of signed digraphs for chemical processes. 2. Control loops and flowsheet analysis, *Industrial and Engineering Chemistry Research*, 42, 4811-4827.

Maurya, M.R., Rengaswamy, R., and Venkatasubramanian, V. (2004). Application of signed digraphs-based analysis for fault diagnosis of chemical process flowsheets, *Engineering Applications of Artificial Intelligence*, 17, 501–518.

Miao, T. and Seborg, D.E., (1999). Automatic detection of excessively oscillatory feedback control loops. *IEEE Conference on Control Applications*. Hawaii, 359-364.

Norvilas, A., Negiz, A., DeCicco, J.., and Cinar, A., 2000, Intelligent process monitoring by interfacing knowledge-based systems and multivariate statistical monitoring, *Journal of Process Control*, 10, 341-350.

Owen, J.G., Read, D., Blekkenhorst, H., and Roche, A.A. (1996). A mill prototype for automatic monitoring of control loop performance. *Proceedings of Control Systems 96*, Halifax, Novia Scotia, 171-178.

Paulonis, M.A., and Cox, J.W. (2003). A practical approach for large-scale controller performance assessment, diagnosis, and improvement, *Journal of Process Control*, 13, 155-168.

Quin, L. (2005). Extensible Markup Language (XML), On-line: http://www.w3.org/XML/ Accessed: 7th Sept 2005.

Rossi, M. and Scali, C., (2005). A comparison of techniques for automatic detection of stiction: simulation and application to industrial data. *Journal of Process Contro*l. 15, 505-514.

Ruel, M., and Gerry, J. (1998). Quebec quandary solved by Fourier transform, *Intech* (*August*), 53-55.

Salsbury, T.I. and Singhal, A. (2005). A new approach for ARMA pole estimation using higher-order crossings. *Proceedings of ACC 2005*, Portland, USA.

Stanfelj, N., Marlin, T.E., and MacGregor, J.F. (1993). Monitoring and diagnosing process control performance: The single loop case, *Industrial and Engineering Chemistry Research*, 32, 301-314.

Tangirala, A.K., Shah, S.L. and Thornhill, N.F. (2005). PSCMAP: A new measure for plant-wide oscillation detection. *Journal of Process Control*. **15**, 931-941.

Tatara, E., and Cinar. A., (2002) An intelligent system for multivariate statistical process monitoring and diagnosis, *ISA Transactions*, 41, 255-270.

Thornhill, N.F. and Hägglund, T. (1997). Detection and diagnosis of oscillation in control loops. *Control Engineering Practice*. **5**, 1343-1354.

Thornhill, N.F. (2005). Finding the source of nonlinearity in a process with plant-wide oscillation, *IEEE Transactions on Control System Technology*, 13, 434-443.

Thornhill, N.F., Cox, J.W., and Paulonis, M. (2003*a*), Diagnosis of plant-wide oscillation through data-driven analysis and process understanding, *Control Engineering Practice*, 11, 1481-1490.

Thornhill, N.F., Huang, B., and Zhang, H., (2003*b*). Detection of multiple oscillations in control loops. *Journal of Process Control*. **13**, 91-100.

Thornhill, N.F., Shah, S.L., Huang, B., and Vishnubhotla, A. (2002). Spectral principal component analysis of dynamic process data. *Control Engineering Practice*. **10**, 833-846.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S.N. (2003*a*). A review of process fault detection and diagnosis Part I: Quantitative model-based methods, *Computers and Chemical Engineering*, 27, 293-311.

Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S.N. (2003*b*). A review of process fault detection and diagnosis Part II: Qualitative model and search strategies, *Computers and Chemical Engineering*, 27, 313-326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N., and Yin, K. (2003*c*). A review of process fault detection and diagnosis Part III: Process history based methods, *Computers and Chemical Engineering*, 27, 327-34.

Xia, C. and Howell, J. (2003). Loop status monitoring and fault localisation. *Journal of Process Control*. 13, 679-691.

Xia, C. and Howell, J. (2005). Isolating multiple sources of plant-wide oscillations via spectral independent component analysis. *Control Engineering Practice*. 13, 1027-1035.

Zang, X.Y. and Howell, J. (2003). Discrimination between bad turning and non-linearity induced oscillations through bispectral analysis. *Proceedings of SICE Annual Conference*, Fukui, Japan.

Zang, X.Y., and Howell, J., (2004). Correlation dimension and Lyapunov exponents based isolation of plant-wide oscillations. *DYCOPS* 7, Boston July 5-7.

Zang, X.Y., and Howell, J. (2005). Isolating the root cause of propagated oscillations in process plants. *International Journal of Adaptive Control Signal Processing*, 19, 247-265