THE SOFTWARE ARCHITECTURE OF A DECISION SUPPORT SYSTEM FOR PROCESS PLANT INSTRUMENTATION

VAZQUEZ G. E., FERRARO S. J., CARBALLIDO J. A., PONZONI I.,
SÁNCHEZ M.C., BRIGNOLE N. B.
Grupo de Investigación y Desarrollo en Computación Científica (GIDeCC)
Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur (UNS)
Av. Alem 1253 – 8000 - Bahía Blanca – ARGENTINA
Planta Piloto de Ingeniería Química (PLAPIQUI) UNS - CONICET
Complejo CRIBABB Camino La Carrindanga km 7 – CC717 - Bahía Blanca – ARGENTINA
e-mail: dybrigno@criba.edu.ar

Abstract: - In this work we present the framework of a new software package to carry out the classification of the state variables in a process plant in order to determine the most convenient instrumentation configuration on the basis of adequate mathematical models of plant behaviour at steady state. We have designed and developed robust and efficient techniques for each step of the categorization task, the key modules being those for observability and redundancy analysis. The individual performance of each module with both academic and industrial examples was successful, revealing comparable or even better results than other existing techniques. Key aspects, such as modularity, user-friendliness, reliability, flexibility, expansibility, functional scalability, and stand-alone capabilities, were taken into account for the design of the integrated software.

Key-Words: - Instrumentation – Decision Support System – Observability – Redundancy – Software Engineering

1 Introduction

The instrumentation analysis of industrial plants basically consists in choosing the most convenient location, quantity and type of the measuring devices to be installed in a plant so as to provide enough reliable information to ensure complete knowledge of the process and its behaviour. In this respect, it is always desirable to make the design economically attractive by tending to select a minimum number of measurements. The central idea is to make use of a steady-state mathematical model that adequately represents the plant in order to calculate as many process variables as possible from a reduced set of strategically located measurements.

Given a certain process, the state variables (such as flowrates, pressures, temperatures and compositions) that have associated sensors that monitor their values are called *measured variables*, the rest being known as *unmeasured variables*. The instrumentation analysis techniques are based on the classification of these variables in order to determine which unmeasured variables can be estimated from model equations as functions of the measurements. The classification of the unmeasured variables is known as *observability analysis* (OA), whereas the categorization of the measurements is called *redundancy analysis* (RA). An unmeasured variable is called *observable* when its value can be calculated from the measurements by means of a model equation. Otherwise, the variable is regarded as *unobservable*. A measured variable is classified as redundant when it can also be estimated as a function of other measurements using model equations.

Various techniques have been proposed to carry out the classification task. A critical review of existing methodologies can be found in [1] and [2]. Depending on the nature of the procedures, two basic approaches can be distinguished. One of them is structural, while the other one is numerical. The former is preferable because the non-numeric features of the analyses makes them independent of the operating points, thus leading to more robust results and wider applicability ranges. Therefore, we designed and implemented robust efficient algorithms based on the structural approach, the core ones being described in detail in [1], [2] and [3].

Instrumentation analysis always requires some sort of direct user interaction so that the resulting sensor layout is tailor-made to fulfil an assortment of requirements, which comprise economic and production objectives, as well as reliability considerations and safety precautions. Therefore, a decision support system (DSS) constitutes the ideal structure for an integrated software package for instrumentation design.

Due to the lack of powerful algorithms and effective well-designed software, the selection of plant sensors has been traditionally carried out on the basis of highly simplified mathematical models that fail to represent process plants in a realistic way. This frequently leads to erroneous classifications, which in turn result in ineffective sensor layouts that tend to provide incomplete knowledge of plant behaviour, which is undoubtedly dangerous. If this risk is overcome by the addition of lots of redundant sensors at random, costs dramatically increase. The availability of the DSS tool described in this work will effectively assist process engineers to obtain trustworthy results efficiently without costly unnecessary overspecification, thus facilitating the complex classification task enormously and leading to significant savings.

2 Design Objectives

The global purpose of this research line is to develop effective tools for process plant instrumentation design that take full advantage of available hardware and software resources in order to achieve more reliable operating policies in modern industry, also taking into account plant economics and safety. In terms of engineering software, our specific goal is to produce a decision support system that provides the most suitable sensor layout through the correct classification of the variables that define the state of a process plant under operation. This paper aims at describing the software architecture of this DSS, including aspects associated with the implementation and testing strategy. The main software features to be required were chosen with the overall purpose of combining agile handling with rigorous analysis to yield a package that is reliable, user-friendly, flexible and expansible.

3 General DSS Framework

The DSS was conceived with the general structure shown in Figure 1, where GM, IM, OM and RM are the acronyms for the Generation, Initialisation, Observability- and Redundancy-Analysis Modules.

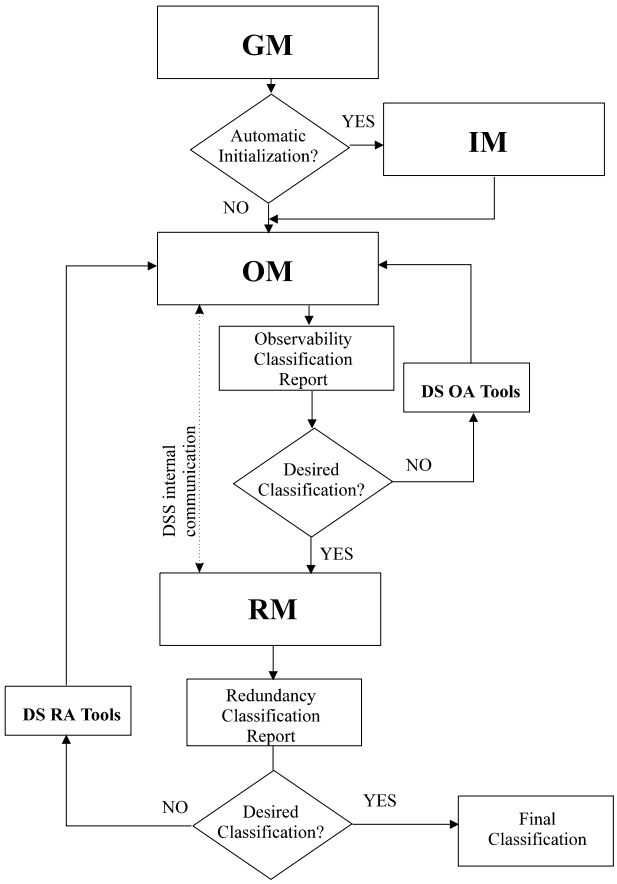

Fig. 1 The DSS Framework.

The DSS executive is the organizer that provides the logical sequence that triggers the execution of all the modules. The role of the executive goes beyond the mere interconnection between routines. It also provides reports of intermediate results at key points of the flow. Those outputs include feasible classification results as well as guidelines generated by the decision support tools for the action to take if the user decides it is necessary to make further improvements in the proposed classification. A typical modification consists in introducing sensors at convenient locations. In this respect, the program predicts the effect of the incorporation of each measurement on the classification results through its decision support tools, thus helping the user determine which sensors to choose in order to yield the desired results in the next iteration of the whole procedure.

4 Software Characteristics

The architecture was planned so that the resulting software exhibited the following useful capabilities:

Reliability: Trustworthy classification results are achieved thanks to the use of a structural non-numeric categorization approach and symbolic analysis tools. The analysis methods can deal with strongly non-linear mathematical models, and the possibility of using rigorous representations, even for complex process plants, ensures correct realistic classification results. The core algorithms can lead to the best structural pattern, which takes full advantage of the mathematical model chosen to represent the process plant under study.

Robustness: A wide range of problems can be solved, including the instrumentation design of complex industrial plants of huge size.

Efficiency: Fast execution is achieved by means of adequate implementation strategies. Algorithmic effectiveness, which aims at achieving high quality results with a minimum amount of iterations, also contributes to run-time reductions. In this respect, the decision-support (DS) tools play a significant role by providing judicious guidelines for the quick convergence to a completely satisfactory final classification.

User-friendliness: The graphical user interface was designed taking into account the standard input-output presentation engineers are familiar with. Results are displayed clearly and the actions to take at the decision-making stages in order to mould the classification results to the specific needs of each user are indicated in a straightforward way.

Flexibility: A modular conception of the software, combined with the introduction of plug-in capabilities, effectively contributes to the generation of a flexible programming environment that is easily adaptable and expandable.

Functional scalability: The general guidelines of the object-oriented programming approach were followed to make provisions for the future incorporation of user-defined models of special items of equipment, additional solvers, symbolic derivation modules, or DS tools.

Stand-alone capabilities: The user can run the modules individually, in case he only wishes to perform a given stage of the analysis.

All the components of the package were conceived so that they complied with these requirements effectively. The role and main features of each module are specified in the next subsections, the modules being described in order of importance.

4.1 Observability-Analysis Module (OM)

This module classifies all the unmeasured variables into observable and unobservable by means of structural analysis techniques. These methods rearrange the occurrence matrix that indicates which variables take part in each model equation to a block lower-triangular pattern that evidences the classification and indicates the sequence of subsystems to solve for the estimation of all the observable variables as functions of the given set of measurements.

The rows and columns of the occurrence matrix, which is always sparse, represent equations and variables, respectively. The observability algorithms permute those rows and columns until convergence to a satisfactory pattern. At the end of the procedure, all the square diagonal blocks that are generated contain the observable variables. Those small dense blocks are called assignment blocks because the diagonal elements indicate which variable has been assigned to each equation. The analysis indicates that the subsystem of equations associated to each diagonal block can be solved separately for its assigned variables, following the precedence order indicated by the general block-triangular pattern.

Two methodologies for OA are available. The user can choose either the GS-FLCN Technique [1], or the Direct Method [3]. The former yields the best classification results for small-size problems because in these cases it is possible to guarantee that the partitioning of the occurrence matrix will contain assignment blocks of minimum order. In contrast, the Direct Method outperforms GS-FLCN for medium- and large-size problems, providing high quality results in significantly lower execution times.

The assignment blocks should be analytically non-singular to guarantee the solvability of the associated subsystems of non-linear algebraic equations. For that reason, the rearrangements proposed by the OA must be validated. The methodology for the detection of singular blocks follows a symbolic approach that is closely related to the RA technique. Therefore, the integration between the OM and the RM naturally arose.

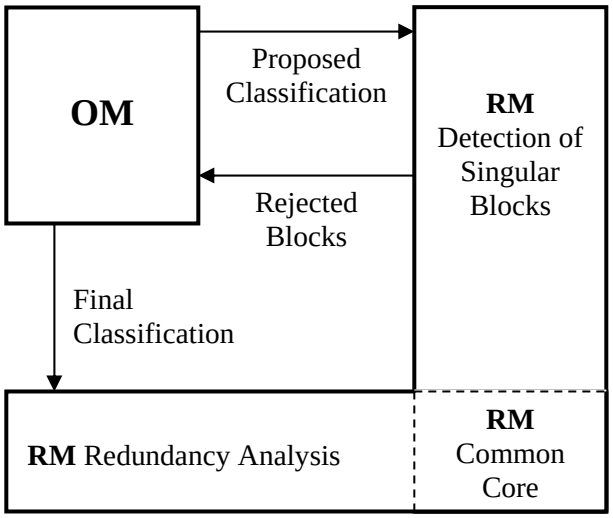

Fig. 2 Interaction between OM and RM.

Both modules communicate as shown in Figure 2. If a singular block is detected, it is tagged as "forbidden" and the observability algorithm is run again. Otherwise, the RM proceeds with the actual RA.

4.2 Redundancy-Analysis Module (RM)

The function of the Redundancy-Analysis Module (RM) is to classify measured variables as redundant or non-redundant. The module is an implementation of the SDIF algorithm (Symbolic Derivation of Implicit Functions) [2]. In addition, the module serves the secondary purpose of detecting singular blocks from the observability analysis as explained above.

In principle, the RM can be requested to classify either a few or all the measured variables in a given system. The RM reads in the full list of equations and measured variables generated by the GM, as well as the output from the OM. After processing, it returns the classification of each variable as "Redundant", "Non-redundant" or "Unknown". When the RM is operating in its secondary mode, i.e. detecting singular blocks, the input is the same and the output is a list of the singular blocks found, formatted as valid feedback for the OM, which will ban the appearance of those blocks from then on.

The module performs many symbolic calculations on algebraic expressions, mainly derivatives and simplifications. It does so by interfacing with an external CAS (Computer Algebra System). One design objective is that the RM be able to use a number of alternative CASes, such as Yacas [4], Maple® (via the OpenMaple® API), or any other package for the same purpose.

For each measurement to be classified, the RM constructs and analyses algebraic expressions according to the SDIF algorithm. This analysis consists in a breadth-first search over a tree of expressions. If no conclusion could be drawn after examining a certain number of tree levels, the variable is labelled "Unknown". In many cases, the "Unknown" status can be removed by deeper exploration or by choosing a different CAS.

The choice of the CAS also influences the speed of the classification, since a very large portion of CPU time is spent on the symbolic manipulation. Internally, the RM addresses speed issues by maintaining expression caches, i.e., lists of expressions that were already simplified or analysed, along with the corresponding result.

Thanks to the adoption of an object-oriented design philosophy, it is not difficult to write a plug-in for a new CAS. Basically, one has to supply methods for computing derivatives, simplifying, handling matrices, calculating determinants, and solving linear systems. The plug-in interfaces directly with the CAS library, which does the actual job.

4.3 Decision-Support (DS) Tools

This module is transversal to the architecture of the whole system. Its objective is to provide intelligent advice that guides the user, helping him to take the most convenient action at the decision-making key points of the classification procedure. The package includes different DS tools for OA and RA.

The main DS tool for OA is based on the concept of decoupling factors [1]. The OA algorithm includes a major loop that yields an intermediate BTF pattern at the end of each iteration. Those results correspond to the best classification that can be achieved with the given set of plant instruments. If those sensors fail to provide enough information, and there are indeterminable variables of interest, the user has to decide where to add sensors in order to make those variables observable without unnecessary over-specification of measurements. The DS tool predicts the effect of the addition of each sensor on the BTF pattern, thus guiding the user to choose only a few additional measurements that will produce maximum increase in plant knowledge in the next iteration.

As regards the RA-related DS approach, the engineer is often interested in having more confidence on the values of certain state variables, which are called key variables. To that end, he could try to ensure more accurate knowledge of those variables by making every key measurement redundant and every key observable variable a function of redun-

dant measurements only. In any case, this can be achieved by incorporating extra measurements. In this respect, the RM assists the user in the judicious choice of those additional sensors.

Finally, the possibility of devising other predictive tools to foresee the impact of the addition and removal of measurements on the basis of the evolutionary approach adopted for the automatic initialisation procedure will also be considered. The central idea is to derive a tool that uses the multi-objective optimisation algorithm that constitutes the core of the initialisation module (IM), thus generating tighter interconnections between the IM and the rest of the DSS.

4.4 Model-Generation Module (GM)

This module builds a system of non-linear algebraic equations that represents the process plant under study at steady state. The mathematical model is generated automatically from the information on plant topology and the initial configuration of sensors. The former is always provided by the user through the Graphical User Interface, while the latter may optionally be generated by the automatic Initialisation Module.

4.5 Initialisation Module (IM)

This module generates a suitable initial configuration of sensors by means of a new algorithm that we generated on the basis of an evolutionary computing approach [5].

The problem of finding an adequate instrumentation layout could in principle be posed as the task of locating a minimum amount of sensors so that maximum knowledge of the state of the process plant is attained during operation. In fact, this is a multi-objective optimisation problem because the solution should provide a satisfactory trade-off among conflicting objectives related to various desirable features of the configuration, such as cost and reliability issues. Since our method was especially devised to serve as the initialisation of the OA and RA techniques employed in this DSS, algorithmic objectives to facilitate the quick convergence to a satisfactory efficient classification in the OM and RM were also incorporated to the formulation of the objective function.

The module contains a multi-objective genetic algorithm whose individuals are binary chains that represent feasible sensor configurations. The procedure judiciously employs classic operators for selection, crossover and mutation in order to favour the survival of the fittest individuals in accordance with a multi-objective criterion. The fitness function follows an aggregated approach for the conciliation of all the individual objectives.

We are now refining the formulation of the fitness function and working on the selection of the most convenient convergence criteria that will constitute the final implementation.

4.6 The Graphical User Interface (GUI)

This module enables the interaction between the user and the DSS. It is basically a window-based interface that provides facilities for the specification of the process units and streams that constitute the plant topology. The user can also locate instruments on any process variable, choose the solvers and specify thermodynamic functionalities. The GUI thus allows the parametrisation of aspects associated to the various modules that conform the DSS, and it is also responsible for the display of all the results and diagnosis information. A complete description of its features can be found in [6].

5 Implementation and Testing Strategy

The detailed design implementation and testing of the DSS was organized in five phases, each of them regarded as the development of a thorough stand-alone software system that would be later extended and evolve towards a bigger software package that in the end would turn into the complete DSS for instrumentation design. As is clear from the description that follows, the division into stages does not necessarily mean that all of them have to be carried out sequentially.

The first phase was the design of the program called ModGen [6], which basically comprises the GUI and the GM described above. At this stage, the software engineering methodology to be employed in the whole project was defined and described in [6] under the name of modified Rapid Application Development (RAD) paradigm.

The second phase consisted in the design and implementation of new observability techniques based on depth-first searches along undirected graphs [1], including the formulation of heuristic rules [7] to guide the search more effectively. Then, the software resulting from Phase 1 was extended to produce a DSS for OA [8]. Later we developed and implemented an alternative method [2] that is ready for integration to the complete DSS.

The third phase refers to the design and development of a new RA method. The RM module described in [2] was conceived, implemented and tested separately and we are at present working on its integration to the software resulting from Phase 2 in order to end up with a package that performs a complete classification of all process variables.

The fourth phase consists in the incorporation of the IM. We have already designed an initialisation strategy [5] and implemented the corresponding prototype. The model-refining task that involves testing algorithmic performance as we make further adjustments on the definition of the fitness function is now well under way.

The last phase of the project is the addition of the DS tools and their integration to the overall system. The algorithm based on the decoupling factors has already been implemented and tested thoroughly, and we are now working on the design and implementation of the DS tool for RA.

In general, all the modules were first developed separately. A prototype for each of the procedural modules was built in Matlab®. Then, the final implementations were written in C or C++. As to the visualization routines, the GUI and the GM were developed in Visual Basic®. Each module, in both prototype and final form, was tested using a suite of study cases of varying size and complexity. In the first place, small academic examples were chosen or designed to provide representative trials. Some of the models were purely mathematical, while others could be associated to standard process items of industrial equipment. Then, the performance on industrial examples corresponding to real plant sectors of increasing size and complexity was attempted. The main three sample cases correspond to a group of mixers and heat exchangers [9], an ammonia plant [10] and an ethane plant [1].

6 Conclusions

We present the general framework for the design and implementation of a complete Decision Support System for the rigorous instrumentation design of industrial plants currently under development. All the main components of the software were designed by our research team and all of them have distinctive features that represent improvements in existing methodologies for the same purpose. The software characteristics make the final product attractive for its industrial application.

References:

[1] Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. A New Structural Algorithm for Observability Classification. *Ind. Eng. Chem. Res.*, 38, 8, **1999**, 3027–3035.

[2] Ferraro, S. J.; Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. A Symbolic Derivation Approach for Redundancy Analysis, *Ind. Eng. Chem. Res*., 41, 23, **2002**, 5692–5701.

[3] Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. A Direct Method For Structural Observability Analysis, *Ind. Eng. Chem. Res.*, **2003**, in press.

[4] Pinkus, A. Z.; Winitzki, S. YACAS: A Do-It-Yourself Computer Algebra System, *Lecture Notes in Artificial Intelligence* 2385, Springer-Verlag, **2002** 332–336. http://yacas.sourceforge.net/

[5] Carballido, J. A.; Ponzoni, I.; Brignole, N. B. Initial Sensor Network Design With A Multi-Objective Genetic Algorithm, *ASAI 2003, (Argentinian Symposium on Artificial Intelligence, XXIII JAIIO)*, Buenos Aires, Argentina, 1–5 Sept., **2003**.

[6] Vazquez, G. E.; Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. ModGen: A Model Generator for Instrumentation Analysis, *Advances in Engineering Software*, 32, 1, **2001**, 37–48.

[7] Ponzoni, I.; Sánchez, M. C.; Brignole, N. B. CDHG: a New Partitioning Algorithm based on the Detection of Cycles in Hypergraphs, *Lat. Am. Applied Res*., 28, Nº1/2, **1998**, 31–36.

[8] Ponzoni, I.; Vazquez, G. E.; Sánchez, M. C.; Brignole N. B. A Computer-Aided DSS for Observability Analysis, *Signal Processing, Communications and Computer Science*, Electrical and Computer Engineering Series, **2000**, 222–227.

[9] Joris, P.; Kalitventzeff, B. Process Measurement Analysis and Validation. *XVIII Congr. The Use of Comp. in Chem. Engng.*, CEF'87, Italy, April **1987**, 41–46.

[10] Bike, S. Design of an Ammonia Synthesis Plant. *CACHE Case Study Report*, Dept. Chem. Engng. Carnegie Mellon University, **1985**.