THE EMERGING TREND TOWARDS KNOWLEDGE-BASED FRAMEWORKS FOR COMPUTER-INTEGRATED MANUFACTURING

G.M. Stanley*
Principal Scientist
Gensym Corporation
10077 Grogan's Mill Road, Suite 100
The Woodlands, TX 77380

KEYWORDS

Computer Integrated Manufacturing (CIM), Knowledge-Based Systems, Software, Process Control Systems, Modeling, Systems Integration

ABSTRACT

Traditional approaches to CIM (Computer-Integrated Manufacturing) involve numerous data interfaces between applications. Some emphasize a centralized database or a hierarchical structure. However, traditional approaches suffer by focusing on data flow: (1) Redundant, possibly inconsistent model information is encoded in multiple applications, complicating development and maintenance, (2) Plant models are not explicit enough for easy review by many people, and (3) Multiple developer and end-user interfaces exist. In reality, there is more commonality between applications than just the data. For instance, multiple applications such as scheduling, control, simulation, monitoring, and diagnostics all need common information, such as connectivity from plant schematics, recipes, manufacturing procedure sequences and constraints, routing information, equipment models, part-of relationships, and goals. Much information about products, equipment, events, and paperwork can be best organized into classification hierarchies, formalizing commonalities. Conventional databases cannot represent this information. A knowledge-based approach to capturing and re-using plant knowledge for multiple applications is required. Knowledge-based systems reduce the gaps between analysis, specification, design, and implementation. Much of the knowledge is in a declarative form, accessible and inspectable by both developers and end-users. Successful examples of computer-integrated operations using newer knowledge-based, network implementations are given.

THE EMERGING TREND TOWARDS KNOWLEDGE-BASED FRAMEWORKS FOR COMPUTER INTEGRATED MANUFACTURING

General CIM tasks and their inter-relationships

CIM (Computer Integrated Manufacturing) is a framework for using computers to manage plants and business at various levels of detail. In the process industries, this is also referred to as CIPO (Computer Integrated Process Operations). We will use "CIM", as the paper applies to all manufacturing.

The general tasks within a CIM framework have often been organized in a hierarchy. The lowest levels of this hierarchy contain sensors, data acquisition and regulatory control. Intermediate levels have tasks for supervisory control, model updating, predictive and "what-if" simulations, adaptive control updates, constraint monitoring, procedure enforcement; fault detection, diagnosis, and recovery; quality control, unit optimization, history-keeping, recipe management, data reconciliation, operator interface, online operator manuals, and other general monitoring/decision support functions. In discrete manufacturing, this includes cell control, dispatching, etc. Higher levels perform plant-wide supervision, coordination between units, plants, and customers, optimization, MRP, routing, scheduling and planning of operations. Ultimately, this is driven by customer needs, technology, and business goals, and is also constrained by government regulations such as environmental restrictions or FDA standards, and other standards such as ISO 9000.

The tasks should interact with each other. For example, high level LP-based or MRP type planning systems generally do not recognize all the constraints present in the plant. When a problem is recognized at the lower levels, there needs to be feedback to high level planning. More examples are given later.

CIM reference models

Reference models for CIM implementation architectures are summarized here: refer to [Venkatasubramanian & Stanley, 1993] for more details. These models have graphical representations: boxes represent computer programs, and arrows in between the boxes represent information flow. The models generally ignore hardware connectivity, although they may be influenced by it. These models emphasize different views of CIM. CIM implementations may include one or more of the following models (views):

(1) Function blocks/ data flow diagrams
(2) Data flow with hierarchical levels ("CIM Pyramid")
(3) Data management
(4) Knowledge management
(5) The network: Client/Server, and distributed object management

(1) Function blocks/data flow diagrams. In this view, specific functionalities such as "data reconciliation", "optimization", "diagnostics", and so on, are broken out and assigned their own boxes in a diagram. The arrows between boxes represent data flow. This model is a called a data flow diagram in traditional software engineering.

(2) Data flow with hierarchical levels. This traditional "CIM pyramid" model organizes tasks based on hierarchical levels. Each level represents a function or set of functions, with data flowing only between

adjacent levels. The CIM pyramid is a special case of a data flow diagram, with additional hierarchical structure and constraints on information flow.

The higher levels are associated with increased geographical scope, reduction in detailed information, and slower time scales. The gaps between the low-level controls and the upper layers of management have rarely been bridged by computer applications in the process industries, leading to the usual "CIM gap".

Traditionally, the layers often coincided with specific hardware. However,
the spread of networking and powerful small computers has "flattened" hierarchies. The hierarchy concept can be useful in reducing system complexity, but is now more abstracted and not tied to hardware implementation. Some hierarchy will probably always be useful. For instance, economic optimization schemes will eventually fail unless sensor validation and general fault detection is handled at a lower level first.

A strict hierarchy can interfere with flexibility and robustness in CIM. Low-level events such as equipment malfunction can have a significant impact on the higher-level planning and control. Flexibility and robustness in dealing with sensor failures or loss of a manipulated variable inherently require multivariable approaches with broader-scoped models and fewer levels. For instance, multivariable optimization-based controls such as DMC do a better job of dealing with these lower-level failures than traditional cascade controls -- they do the best they can with the number of sensors and the degrees of freedom available.

As another example, fault detection to recognize sensor failures generally sits at a level "above" the regulatory controls, since it needs to analyze data from more than from just one controller. However, once a sensor has been declared "bad", related lower level controls should be disabled. Similar considerations apply to the relationship between data reconciliation and fault detection. Fault detection schemes associated with data reconciliation can help identify faults. Other, separate fault detection schemes might also detect failed sensors or equipment (for instance, by looking at valve positions and controller windup). In either case, once a sensor is declared "failed", the normal data reconciliation should be reformulated to ignore the failed sensor or equipment. There should be peer-to-peer communication between the subsystems associated with fault detection and data reconciliation.

(3) Data management view. Here, the center of the system is a database. Applications all tie to this core, only passing data back and forth to the database (and not to each other). Hierarchy is not represented in this view, although there may be a link drawn to company headquarters. Traditional software engineers emphasize "data modeling" in designing these systems. The focus is on the data, independent of its usage in different tasks.

One strength of the data management model is in enforcing data consistency. One set of data is agreed to by everyone. A problem caught by one system such as a diagnostic system, can be corrected, and corrected data can be used by the other systems, and for historical analysis. In practice, low-level functions that must operate at relatively high speed may get data directly from the DCS or PLC, bypassing the slower database. However, the database remains the repository for historical information.

If all communication between applications is through the central database, this reduces the number of possible interfaces, compared to function block approaches. However, it also limits the quality of the communications between the applications, since all that can be passed is simple data. Queries of one application to another regarding underlying models are not defined, difficult to implement, and depend on side-effects of setting of database values.

A major problem with the data management view is that it doesn't recognize the common models needed across applications. As it is based on earlier database technologies, the databases are not well suited for storing models, schematics, complex relations between individual objects or classes of objects, etc. However, most CIM installations use, and will continue to use, a database for maintaining historical data.

(4) Knowledge management view. The knowledge management view of CIM emphasizes common knowledge for multiple applications. A major difference from database-centered approaches is that the common information includes not only data, but also contains representations of plant equipment, their models, and operating rules and procedures. For instance, data reconciliation is just a mathematical program that accesses the common representation used by all the applications. It shouldn't contain its own, independently entered version of the process topology used to generate the constraints (balance equations).

The knowledge management approach generally implies an object-oriented representation. There is an emerging consensus that the object-oriented paradigm is the best way to manage inherently large, complex, heterogeneous software collections like CIM.

The emphasis on a common core of knowledge, especially models in an object-oriented form for multiple applications in control, diagnosis, and other areas, is exemplified by [Årzén, 1990, 1991], [Stanley, 1991], [Moore & Stanley, 1993], [Hoffman, Stanley & Hawkinson, 1989], and [Terpstra, Verbruggen & Bruijn, 1991]. G2, the real-time expert system from Gensym, is the most widely used tool for knowledge-based systems integration. It has particular strengths in its close link between objects and their graphical representation, and in external interfaces needed for CIM.

Knowledge-based products include powerful tools for representing plant models, such as inheritance in a class hierarchy. This simplifies development and maintenance by abstracting common properties and behaviors. For instance, a tank and a warehouse have some similar attributes such as inventory, and some similar behaviors, such as material balance. A "superclass" called "container" can be created with an inventory attribute and a material balance behavior. Both the tank and warehouse automatically inherit these properties, modeled as "a-kind-of" container. A tank may have further specializations in the class hierarchy, such as vertical drum, horizontal drum, etc., which differ only in the relationship between level and inventory. Corrections in models then only need be entered at one spot in the hierarchy, simplifying development and maintenance. Other types of relationships, such as "part-of", "uses", "connected-to", "is-at", "customer-of", and so on, are also modeled explicitly and graphically, so that the knowledge can easily be reviewed by a large number of people.

Knowledge-based plant models may be steady-state or dynamic, qualitative or quantitative. Often, any needed equations can be generated automatically from higher-level models, such as schematics. The model information includes generic equipment descriptions, specific attributes such as equipment sizes,

interconnections such as pipes and wires between the equipment, standard procedures, mathematical and qualitative models of the equipment, diagnostic techniques for that equipment, and so on. This view encourages high-level plant descriptions such as schematics, and multiple uses of models.

Traditional relational databases are inadequate to hold this information. An object-oriented representation, complete with graphical knowledge representation, is required.

A key factor is that the same information is used in multiple applications. Going beyond the data consistency of the database systems, there is consistency in the plant model used by all the applications. This "model consistency" is a major advantage in reducing the amount of development and maintenance effort to keep up with plant changes. You only need to build or change the plant representation once, rather than in each separate application.

Graphical languages to define the discrete steps needed in discrete manufacturing, batch operations, or in startups and shutdowns, have become popular. These are built as knowledge bases, and given names such as "Procedure Management Systems", or "Sequential Function Charts". These techniques are used in some of the case studies, and have been built into commercially available products such as Gensym's GDA, which combines this sequential control with data flow diagrams [Stanley et. al, 1991], [Finch et. al., 1991], [Fraleigh et. al, 1992]. These have the advantage that multiple applications can easily analyze the diagrams. For instance, the representation of steps in a control sequence could be used for scheduling as well. Another graphical representation technique, designed to complement the normal flowsheet information with "means-end" information, for multiple applications such as diagnosis and planning, is the "multilevel flow modeling (MFM) approach [Larsson, 1992].

A benefit of the knowledge-based approach is that it reduces the gaps between system analysis, specification, design, implementation, run-time use, and maintenance. Much of the model, such as the class hierarchy, is carried through from analysis to implementation and maintenance. More of the information is in an explicit, "declarative" form, accessible and inspectable by both developers and end-users, rather than buried in procedural code. Thus, mistakes can be more easily caught. Inconsistencies can be resolved, and there is just one place where each given type of knowledge is represented. For instance, in the case of the schematic representation, everyone from plant operators to engineers to managers can easily notice mistakes. This contrasts to traditional programming, hidden inside of function blocks in the CIM diagram, where only programmers could detect errors and understand the representation.

Knowledge-based tools offer additional implementation benefits. They are typically portable, so that applications developed on one machine can be moved to another. The applications are not tied to a particular control vendor. The applications can outlast the lower level controls, and migration paths to new hardware can be simplified. The applications can also more easily be changed, within security limits established at the site. The best tools are also distributable over networks, allowing multiple users from different consoles, each with their own interface and access rights. The knowledge bases can also communicate with each other, so that distributed applications can be built - scaling up well as the size of the plant increases. The knowledge bases also can communicate with other conventional applications.

(5) Network view: Client/Server, and distributed object management
The spread of networks has radically altered computing and CIM. Every piece of hardware, and every application, can potentially talk directly to each other as peers. Local area networks cover most plants now; high-speed wide-area networks can link sites more closely than before. One effect has been to flatten hierarchies, as already noted.

In contrast to the data management view, there is no single "core" program. Rather than a "star" topology with a database at the core, the network itself is the heart of the system. The network view has a close synergy with an object-oriented view, since network elements communicate by exchanging messages.

Unlike the data management approach, the network approach allows the full connectivity between applications of the functional module approach. However, the number of interfaces to be written is just the number of applications. The reason is that each application just needs to interface to the network, rather than having a set of pairwise interfaces. Each application responds to messages, regardless of who sent them. This is simpler and more flexible than a central database approach.

Client/Server architecture is an outgrowth of the network view. Any program can act as a server, and any client (user) can access the programs over a network. "Servers" specialize in functions such as history-keeping, file-serving, or numerical computation. Other servers provide access to particular variables by tagname, independent of their physical location on the network. Networked windowing interfaces such as X-windows follow client/server architecture.

For CIM, there can also be servers for plant model representations; functions such as optimization; managing and logging of messages between people, and managing and logging of messages from applications. As an example, if an instrument has been repeatedly causing transient problems, a message can be electronically mailed to a control engineer, instrument engineer, or other system maintainer, advising them of the recurring problem.

The evolving Fieldbus network standard will further flatten CIM architecture by allowing communications directly from high-level programs to smart sensors in the field, bypassing today's conception of a DCS. Smart sensors will recognize and transmit some of their own failure modes, improving diagnosis by helping to better distinguish between sensor problems and process problems. This improved reliability will also improve the success rate of other applications depending on the data, such as online optimization and model updating.

The need to go beyond mere "data integration" for CIM
Traditional CIM systems pass data between the various tasks. This "data integration" often is centralized in a relational database, or a historical database specialized for keeping historical time-series and event data.

For a comprehensive system, the central data repository needs to represent the entire state of the plant over time. This includes not only plant variables, but also includes many statuses. For instance, for effective fault diagnosis in the presence of feedback controllers, the automatic/manual status, the local/cascade status, and the anti-windup status of controllers is required. This information is needed because the expected system behavior, visible symptoms, and propagation of faults can be drastically

different depending on those statuses. Similarly, design data may be needed. For higher-level tasks, data is needed on customer orders, shipments, maintenance records, etc.

However, there is a deeper level of integration that should be recognized. There is common information needed by many of the tasks listed above. For instance, the process schematics, equipment parameters, and design equations are needed in many applications, such as optimization, control, diagnosis, data reconciliation, leak detection by material balancing, simulation for design or operator training, user interfaces, controller tuning, and other tasks. A graphically-oriented, object-oriented system captures schematics, along with equipment attributes and associated models.

Controller configuration, or sequential control strategy, is needed for fault diagnosis as well as control. Then expected behavior can be derived, and hence deviations from expected behavior or plans can be recognized. The same is true of recipes, possible routings (product processing sequences), and schedules: Scheduling, sequential control, and scheduling share this information. In addition, fault diagnosis systems need this information to detect deviations from expectations. More generally, the applications depend on common plant and business models at various levels of abstraction.

It is inefficient and error-prone to enter the same model information into each application independently. Furthermore, maintaining redundant model information within each application leads to a software maintenance problem. As long as CIM is basically viewed as a collection of independent programs with data interfaces, it will be expensive to build and maintain the systems. Central relational or specialized historical database do not solve the problem, as they are poor at representing much of the model information that is needed.

The plant and business models play a central role in these applications. There are also other uses for these models. For instance, there may be process design simulations for performing what-if scenarios, and operator training simulations. The newer CIM-based applications recognize common plant model structures. Thus, instead of just integrating data as in traditional CIM, the models are available within the integrated framework as well.

The earlier CIM reference models may help in conceptualizing and designing CIM systems, but the actual implementations are following general trends in software. They are starting to move to object oriented systems, distributed over networks (viewpoints 4 and 5 above). Databases will likely continue to record simple historical data, but they will be server nodes on the network, rather than the center of the system.

End-user and developer interfaces in CIM

Operator and developer interfaces are extremely important and time-consuming to develop, yet are almost never shown in the theoretical reference models. The reference models are mainly oriented towards programmers, not the end user. The end-user may see the systems quite differently, and have trouble with the function block abstractions visible mainly inside the computer codes. Also, the end users and developers both may have trouble learning the different interfaces associated with separate functional programs.

End users and developers prefer as few interfaces as possible. There is often resistance to installing too many screens. X-windows allow multi-functional use of CRT screens and remove the hardware

inconsistency issues. However, each software application, when developed separately, could have different, inconsistent interfaces for both end-users and developers. This creates an overall CIM system that is hard to learn and use, where everyone can too easily make mistakes. An important part of the usability of the overall CIM system is a common style of users' interfaces.

Users want the interfaces easy to use, yet powerful: summarizing large amounts of information graphically, on demand with minimal keystrokes, and reporting on an exception basis with intelligently-filtered alarms. The trend towards "operator empowerment" accentuates the need for better user interfaces. The operators are taking on more of the roles previously handled only by supervisors and engineers.

The developer and end-user interfaces significantly affect development and maintenance costs. It is common that operator and developer interfaces, along with data interfaces, consume 70% or more of the entire development effort. This development time is shortened by using a powerful knowledge based graphical interface.

Many have hoped that basing CIM on a relational database would solve their interface complexity problems. Databases do provide a standard interface for programmer access to data. They also may provide a standard query approach for end-user access to current or historical data. However, this does not guarantee that the applications have any user or developer interface consistency. It also doesn't provide access to the more complex data structures associated with the applications, such as the schematics and relationships actually used to generate material balances or diagnostics.

These needs favor the use of graphical knowledge-based tools for CIM systems integration. Existing DCS interfaces are not adequate for the task, and stand-alone graphics packages require too much specific programming for each application. The representation ability of knowledge-based tools is powerful enough to support the development of many diverse applications, yet still present a single graphically-oriented interface configurable for the needs of both end users and developers. For example, [Ilgen, 1993] describes a knowledge-based operator interface for advanced controls. Additionally, the ability to share some of the same representations between developers and users improves the confidence of the users in the system, and reduces the errors. The common graphical interface, tied in with universal access due to the spread of networks, is simplifying the installation and acceptance of CIM.

Case studies

There are now numerous successful CIM projects that address multiple applications using the "knowledge management" approach. These projects generally combine elements of the data management, knowledge management, and network views. These projects also tended to stress the importance of good, consistent end-user interfaces for acceptance of the systems. Similar considerations are generally true for the developer's interface as well; the most successful applications generally have a significant overlap between the two interfaces. A number of successful industrial applications using knowledge-based systems are reviewed by Stanley [1991], and [Venkatasubramanian & Stanley, 1993].

A food-processing plant in Australia has created an extensive CIM system [Dye, 1991; Food Processing, 1992]. The goal was a fully integrated, paperless computer system to control and manage production, starting with raw material input, and continuing through ingredient preparation, processing, packaging,

and warehousing. Minimal staffing was desired and achieved. Real-time information sharing between the plant floor and upper management was seen as a key to success. The system is based on PLC's, databases, and a graphical, real-time expert system shell for integration, operator interface, monitoring, control, and other applications. A network provides access to the entire system from any of the multiple engineering workstations.

The system does scheduling, and advises the operator of the current products scheduled to be run, based on production orders and recipes stored in the database, current materials on hand, and so on. Once the operator accepts the scheduled run, the system then coordinates the entire production, including retrieval of processing condition setpoints, limits, alarms, and other data, setup of controls, sequence control by sequential function charts, optimal resource allocation, and coordination between the various stations.

A comprehensive, knowledge-based system has been installed online at a sugar beet plant of ISI Agroindustriale Sugar Company in Italy (Filippini et. al, 1990). The system advises operators in the following areas: evaluation of regulatory control loops, prevention of malfunctions, standardization of quality and yield, and optimization. The system has been well received, and is considered an essential tool in plant management. British Sugar has independently developed a similar system for its beet sugar plant at Ipswitch in the UK [AI Intelligence, 1992], [Cheng & Daldry, 1992].

Eli Lilly and Company has developed knowledge-based systems for operation and control of industrial fermentation, both in manufacturing and in pilot plants [Alford et.al., 1992]. Applications include online data validation, process analysis, operator interface, reduction of nuisance alarms, optimal nutrient feed time prediction, and remote alarming. High-level situation analysis tracks trends, for instance, in carbon dioxide evolution, to plan for the next steps in a batch process. Similar applications have been reported for pilot-scale plants by Novo Nordisk at a pilot plant near Copenhagen, and the University of Newcastle [Aynsley et. al, 1989, 1990].

Air Products and Chemicals has developed a comprehensive, standardized knowledge-based CIM application for their air liquification/separation plants ("Air Separation Units") [Cartledge et. al., 1991], [Cartledge, 1992], [Venkatasubramanian & Stanley, 1993]. One knowledge-based system serves as the closed-loop supervisory control system, optimization system, training system, and operator interface for the entire plant. The system also diagnoses states of the plant and determines the best course of action. The primary justification is energy efficiency. Part of the philosophy is trying to consistently achieve the results of the best operator. Considerable attention was paid to the operator interface design. This system has been installed at more than a half a dozen sites, and there are plans for more.

There are many businesses where companies have plants with a high degree of similarity all across the world. For these companies, it is important to standardize applications, to minimize support costs, and consistently achieve the best possible results from all the plants. The knowledge-based approach to CIM has been very fruitful for these companies, because the representation techniques allow easy modeling of minor differences through "subclassing", and through simple, graphical configuration of objects. The Air products application is supported from a central headquarters - there are typically no engineers dedicated to the sites. In some plants, operators initiate their own additions to the knowledge-based application, with consulting over the phone to their centralized support. The system is designed to customize those aspects, while standardizing commonalities such as process configuration. Similarly,

Lafarge is standardizing cement plant automation for 60 worldwide plants via knowledge-based techniques [Lermant, 1993].

An example of the linkages between various CIM applications is at the Thorold, Ontario plant of the Quebec and Ontario Paper Company [Harvey, 1992]. Their system combines scheduling, inventory control, simulation, and supervisory process control, to minimize costs. The authors also highlighted the importance of the use of an object-oriented, graphics oriented system for development. Not only did it shorten development times, but also allowed for better buy-in by operators, who could directly inspect a large portion of the system.

Another example combining monitoring, simulation, and operator-in-the-loop control in a paper mill is the EPAK system, offered commercially by ABB AFORA of Finland, and installed at various sites. EPAK provides decision support to the operators in a paper mill run by Norske Skog AS of Norway (Yeager, 1990; Opdahl, 1989). The incentives are to improve quality, minimizing variations between shifts. This application also highlighted the importance of operator review of the models.

[Moore and Stanley, 1993] summarize a variety of knowledge-based applications with models playing a central role for multiple applications. In addition to end-user examples for aluminum plants, solid fuel propellant manufacturing, and chemical incineration, there are applications for sale by traditional engineering, control, and simulation companies. There is a description of Setpoint's PROCISE scheduling system with graphical model entry and simulation, and Simulation Science's ROM (Rigorous On-Line Modeling). The ROM framework is designed for entry of the common plant model in a knowledge base, with subsequent online use for data reconciliation, control, optimization, troubleshooting, training, and general management decision support.

A similar general-purpose knowledge-based framework for multiple model-based applications is under development by ABB Simcon in New Jersey, USA [DeHaan et. al., 1993]. Another example of using a common model is provided in [Stanley, 1993], where piping schematics and associated generic equations are used to generate simulation solutions, data reconciliation solutions, and to generate cases for a neural net-based diagnostic system.

REFERENCES

AI Intelligence, "Sugar Producers Apply KBS to Beet Sugar Production", AI Intelligence, August, 1992, pp. 9-10.

Alford, J.S., G.L. Fowler, R.E. Higgs, Jr., D.L. Clapp, and F.M. Huber, "Development of Real-Time Expert System Applications for the On-Line Analysis of Fermentation Data", ACS Conference Proceedings Series: Harnessing Biotechnology for the 21st Century, Ed. M.R. Ladisch and A. Bose, 1992, pp. 375-379.

Årzén, Karl-Erik, "Knowledge-Based Control Systems, American Control Conference (ACC-90), San Diego, 1990.

Årzén, Karl-Erik, "Knowledge-Based Applications in the Process Industry: Current State and Future Directions, IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control, Bergen, Norway, May 29-30, 1991.

Aynsley, M., D. Peel, and A.J. Morris, "A Real-Time Knowledge-Based System for Fermentation Control", Proc. ACC, Pittsburgh, USA, 1989, pp. 2239-2244.

Aynsley, M., A.G. Hofland, G.A. Montague, D. Peel, and A.J. Morris, "A Real-Time Knowledge-Based System for the Operation and Control of a Fermentation Plant", Proc. ACC, San Diego, USA, 1990, pp. 1992-1997.

Cartledge, J.G., "A Plant Operator Interface for G2 Process Applications", Gensym Users Society proceedings, Orlando, Florida, USA, March 11-13, 1992.

Cartledge, J. , M. Eddinger, R. Elward, L. Krieger, S. Salvo, D. Vinson, "Closing the Loop: Optimizing Control of an Air Separation Facility Using a Real-Time Expert System", AIChE Annual Meeting, Symposium 134, Nov., 1991.

Cheng, P., and S. Daldry, "IDES: Expert System for Diagnosis and Decision Support", CITS Meeting, Finland, May, 1992.

DeHaan, S., Y. Jain, M. Kutten, P. Morrow, "A Unified Approach to the use of Simulation for Process Plants", in On-Line Real-Time Simulation in Advanced Control Systems, Proceedings of 19th Purdue Advanced Control Conference, August 30-Sept. 1, 1993, Lafayette, Indiana, USA, pp. 157-165.

Dye, Ross, "Expert Systems in CIM Applications", Process & Control Engineering (PACE) (Australia), Vol. 44, No. 11, November, 1991, pp. 36-38.

Filippini, F., S. Urbinati, and M. Solimano (1990). An Expert System for the Prevention of Malfunctions and Technological Optimization in a Beet Sugar Plant. Proc. European Gensym Users Society Meeting, Munich, Germany, Dec., 1990.

Finch, F. Eric, G.M. Stanley, and S.P. Fraleigh, "Using the G2 Diagnostic Assistant for Real-Time Fault Diagnosis", European Conference on Industrial Applications of Knowledge-Based Diagnosis, Segrate(Milano), Italy, Oct. 17-18, 1991.

Food Engineering Magazine (Special Supplement), Nov. 1992.

Fraleigh, Steven P., F. Eric Finch, and Gregory M. Stanley, "Integrating Dataflow and Sequential Control in a Graphical Diagnostic Language", IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries, Newark, DE, USA, April 22-24, 1992.

Harvey, D.M., "New Opportunities for Supervisory Process Control", Control Systems '92, sponsored by the Tech. Section, Canadian Pulp & Paper Association (CPPA), Whistler, British Columbia, Canada, Sept., 1992.

Hoffman, A.G., G.M. Stanley, and L. B. Hawkinson, "Object-Oriented Models and their Application in Real-Time Expert Systems", Simulation and AI 1989, Simulation Series, Volume 20 Number 3, Proc. Society for Computer Simulation International Conference, San Diego, Jan., 1989.

Ilgen, M., "Expert System as an Operator Interface for Advanced Controls", Proc. Industrial Computing Conference (ICS/ISA), Chicago, Illinois, USA, 1993, pp. 265-276.

Larsson, J. E., "Model-Based Measurement Validation using MFM", On-line Fault Detection and Supervision in the Chemical Process Industries - IFAC Symposium, Newark, Delaware, USA, April 22-24, 1992, preprint pages 87-92.

Lermant, N., "A Two Years Experience with G2 to Automate Cement Kilns", Gensym European User Meeting, Leuven, Belgium, Oct. 1993.

Moore, R.L., & G.M. Stanley, "Integrating Simulations With Real-Time Expert Systems, in On-Line Real-Time Simulation in Advanced Control Systems, Proceedings of 19th Purdue Advanced Control Conference, August 30-Sept. 1, 1993, Lafayette, Indiana, USA, pp. 75-85.

Opdahl, P., "EPAK - Expert System for Paper quality", Gensym User Meeting, Fall, 1989.

Stanley, G.M., "Experiences Using Knowledge-Based Reasoning in Online Control Systems, plenary paper presented at International Federation of Automatic Control (IFAC) Symposium on Computer-Aided Design in Control Systems, Swansea, UK, July, 1991.

Stanley, G.M., "Neural Networks for Fault Diagnosis Based on Model Errors or Data Reconciliation", ISA 93 (Instrument Society of America), Chicago, IL, USA, Sept. 19-24, 1993.

Stanley, G.M., F.E. Finch, and S.P. Fraleigh, "An Object-Oriented Graphical Language and Environment for Real-Time Fault Diagnosis", European Symposium on Computer Applications in Chemical Engineering (COPE-91), Barcelona, Spain, Oct., 1991.

Terpstra, V.J., H.B. Verbruggen, P.M. Bruijn, "Integrating Information Processing and Knowledge in an Object-Oriented Way", IFAC Workshop on Computer software Structures Integrating AI/KBS Systems in Process Control, Bergen, Norway, 1991.

Venkatasubramanian, V., and G.M. Stanley, "Integration of Process Monitoring, Diagnosis and Control: Issues and Emerging Trends", FOCAPO-2 (Foundations of Computer-Aided Process Operations II), Crested Butte, Colorado, USA, July 18-23, 1993.

Yeager, R., "Norske Skog Uses Expert System to Optimize Production Quality", Pulp & Paper, Dec., 1990.

Presented at Instrument Society of America ISA/94 Conference, Philadelphia, Pennsylvania, USA, May 1-5, 1994.

* Contact the author at http://gregstanleyandassociates.com/contactinfo/contactinfo.htm