Carl Stern, SandiaView Software, Inc. & George Luger, University of New Mexico

Integrating Model-Based Diagnosis and Control Automation

Abstract

This paper describes the development of a new knowledge-based framework for control automation and diagnosis. This framework relies on several key technologies: a knowledge-based hierarchical control architecture, an object-oriented human-computer interface, embedded model-based control, and automated model maintenance. The role of each of these technologies in an integrated system for control automation is discussed. It is further shown how an automated model-based control framework provides a unique range of opportunities for the application of automated diagnostics, and conversely, how automated diagnostics can contribute to a control system that is robust and adaptive.

Overview

Vista Control Systems, SandiaView Software, and the Artificial Intelligent Research Group at the University of New Mexico are collaborating on a series of research projects to develop a new approach to control automation. The framework for this approach, involving a knowledge-based hierarchical component architecture has been developed and tested in the domain of particle beam accelerators. Recent work has focussed on embedding model-based diagnostics and process monitoring within this framework.

The modular design of this control framework reflects the convergence of several key software technologies: object-oriented programming, hierarchical component-based integration, and model-based control. Sets of individual controllers are organized hierarchically in a framework in which higher level controllers monitor and coordinate the actions of lower level locally targeted controllers. Object-oriented abstraction is used to provide each level of the control hierarchy with a view of the data that reflects the nature of the interests and actions of controllers at that level.

Finally, a model-based framework structures the hierarchical decomposition of the plant into locally controlled subsystems. The correspondence between the organization of the control hierarchy and the model-based representation of the plant allows a natural implementation of model-based control, process monitoring, and diagnostic techniques.

This research began with the design of control systems for particle beam accelerators. This is a challenging task because it involves control of large machines with a variety of component subsystems and complex interactions.

In the following section, discussion begins with a brief introduction to the nature of the accelerator control problem and the design criteria for this project. Next, key dimensions of the control architecture are presented. These include use of a hierarchical framework for coordinating local controllers and application of a knowledge-based approach to control. This knowledge-based approach, implemented primarily through a teleo-reactive execution and planning mechanism, is then described.

Finally discussion focuses on integration of model-based control and the development of an automatic system for model maintenance and recalibration. In the last few sections, test results of this accelerator control architecture are presented and their potential implications for the development of large scale distributed applications is briefly discussed.

Design Criteria for Accelerator Control

Particle beam accelerators present an interesting test bed for control automation technology. Control problems in this domain are typically of significant size. Furthermore, the potential for global propagation of local effects complicate naive attempts to decompose control into smaller more localized domains.

Stern, CR. And Luger, G.F. "Integrating Model-Based diagnosis and Control Automation". In *Proceedings of the Intelligent Ships Symposium III*. Philadelphia PA: ASNE Press (1999).

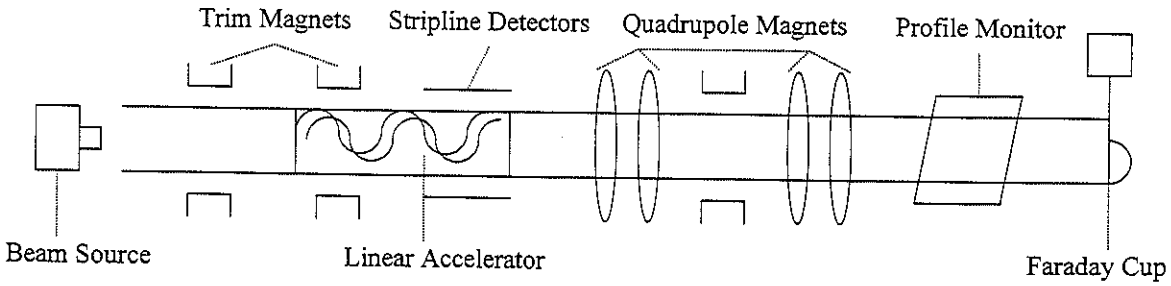

Figure 1. An accelerator section.

Human operators generally coordinate the tuning operations required to achieve optimal machine performance. These tuning operations can be quite time consuming and expensive, reducing the time available for end user experiments. While models are used to derive initial control settings, calibrating and maintaining these models is also time consuming and resource intensive. Thus successful control automation technology in this domain has the potential for important reductions in cost and increased efficiencies in machine utilization.

Vista's development of a framework for automating accelerator control operations has been guided by a few key design considerations. The framework needed to be modular and extensible so that new functionality could be added incrementally. The design needed to be general and portable, so that a customized control system could be deployed at a new facility in a reasonable period of time with a modest level of effort. Finally, the tuning and diagnostic methodologies implemented in software needed to be similar in nature to those currently in use by human operators, so that only a modest level of training would be required to enable existing operations staff to supervise and support this new control software.

These first two design criteria have been satisfied through the development of a hierarchical component-based control architecture. The last criterion is addressed through the use of a knowledge-based approach. A representation recently developed at Stanford [Nilsson 1994], called teleo-reactive control, has been adapted to implement in software the control logic currently used in tuning operations. Both of these design elements will be discussed in more detail in the next three sections.

Once this initial control automation framework was developed, effort has focussed on the integration of model-based diagnosis and control. Embedding control models in the control system itself offers the possibility of dynamically refining and recalibrating models interactively to improve system performance.

Equally important, online models provide the opportunity for model-based diagnosis and system testing. The accelerator model is used to generate predictions that are compared to observed system function. Discrepancies between predictions and observations are analyzed in order to identify their source in modeling errors or machine malfunction. Work in developing automatic model calibration and diagnostic components is discussed in considerable detail later.

The Accelerator Control Problem

Figure 1 shows a typical accelerator beamline, including a beam source, trim magnets for steering, an accelerator, quadrupole magnets for focusing, Faraday cups and stripline detectors for measuring current, and profile monitors for measuring beam size and position. Various components are placed along the beamline by design to produce specific effects in a known way.

Unfortunately, real systems rarely work as they are designed. Challenges arise from variations in beam production, remnant magnetic fields, poorly modeled beam behavior, misplaced or flawed control elements, and changes to the design or use of the facility after it has been built. For example, energy spread, emittance, and dispersion at the source determine the sensitivity and effectiveness of steering and focusing operations further down the beamline. Steering and

focusing operations interact with each other due to the steering effects of focusing magnets and the dispersion effects of bending magnets.

Beamline designers consider these problems and build diagnostic components into the beamlines. The accelerator model provides initial parameter settings. Profile monitors, position monitors, current detectors, etc., measure actual beam parameters throughout the line to provide information for verifying or correcting beam characteristics. Human operators use these diagnostics to interactively tune the beamline, alternating between source tuning, steering, and focusing operations. Automating this highly adaptive data driven behavior of human operators presents a significant challenge.

An Hierarchical Component-based Framework

The accelerator control system is based upon a distributed hierarchical architecture intended to incorporate a wide variety of representations, both analytic and knowledge-based, into a single control framework. At the heart of the architecture is a group of knowledge-based *controllers*. These controllers are hierarchically organized in a structural/functional hybrid design.

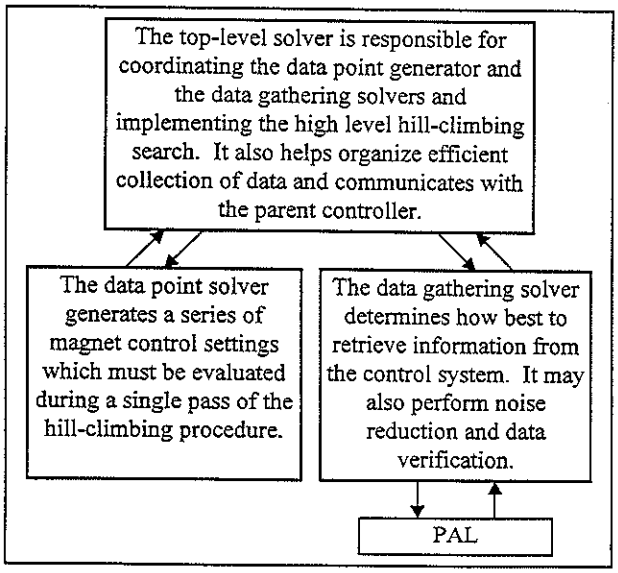

Figure 2. A set of components for hill-climbing optimization.

Controllers are responsible for making decisions about what control actions will be performed, when they will occur, and how their performance will be measured. Controllers are also responsible for reasoning about system state, diagnosing errors in control solutions, decomposing goals into tasks and actions, and initiating any necessary human interaction.

Controllers carry out plans that accomplish user-defined goals by applying various forms of domain knowledge. *Solvers* are reusable components that can be configured by controllers to apply low-level, well-defined algorithms to the control process.

Solvers encode procedures that can be assembled (again in a hierarchical manner) for run-time construction of control algorithms. Figure 2 shows an algorithm, in this case simple hill climbing. The typical *solver*-based procedure for optimization is broken into three parts: a *solver* for generating data points which must be measured during search, a *solver* for measuring the data points using appropriate elements in the domain, and a parent *solver* which coordinates actions of the two children in a way that performs hill-climbing. A different algorithm, Newton's method for example, can be constructed by merely changing the top-level procedure in the parent controller. Different *solvers* may also be substituted at the lower level to handle specific constraints, e.g., specialized noise handling to handle certain kinds of oscillations in the source.

Because a symbolic system for reasoning about the control system is used, raw data is rarely appropriate for direct manipulation by controllers. The same is true in reverse; a low-level interface for manipulation of control elements is usually inappropriate. For this reason an object-oriented Physical Access Layer (PAL) has been developed as an abstraction mechanism between controllers and the underlying control system. This provides a number of important advantages:

- The PAL provides a mechanism for hiding unimportant implementation details about hardware and provides a uniform interface for control access.

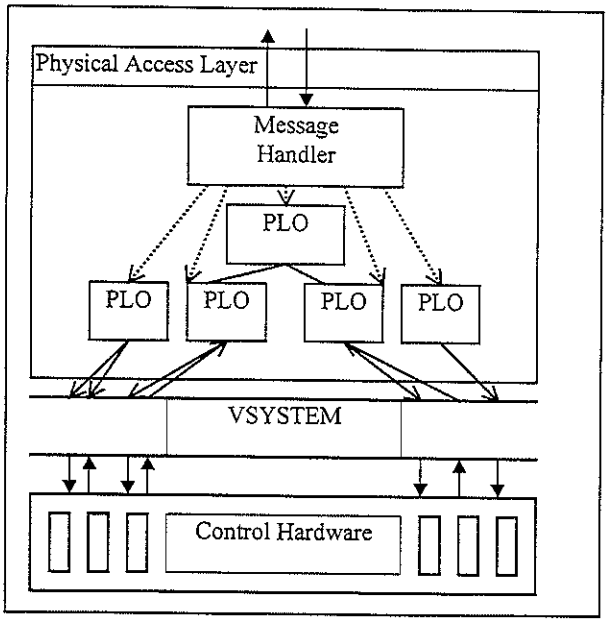

Figure 3. The Physical Access Layer is composed of abstractions called Physical Layer Objects.

- Resource conflicts can be initially handled at a low level and, once identified, mediated at the controller level.
- Controllers can pass filtering instructions to the PAL to allow pre-processing of data into a representation expected by a controller. This can happen, for example, by giving the PAL fuzzy sets for classifying data, or passing a neural network encoding to the PAL.
- The system is highly portable. By abstracting underlying control elements, control algorithms can be written in a generic manner. The same set of controllers can be used at multiple accelerator facilities by changing the low-level data handling in PAL.

The PAL is composed of a number of Physical Layer Objects (PLOs) which are representations of a control or diagnostic element or collection of elements. These objects can be as simple as single magnets, or as complex as non-linear tuning knobs that manipulate a series of magnets. PLOs communicate with hardware indirectly through Vsystem, a high-speed software data bus (Clout 1993). PLOs can be organized in a hierarchical fashion and operate in parallel, much like controllers. The PAL provides PLOs access to a library of tools for representing and filtering data, algorithms for noise handling, pattern recognition, and feature extraction. Figure 3 illustrates the design of the PAL.

The PAL does more than provide a high-level interface to the underlying software control system. The PAL also performs low-level control over groups of components that together represent a control or measurement element. For instance, at Brookhaven National Laboratory's Accelerator Test Facility (ATF), beam measurements are usually taken through profile monitors, which consist of phosphor screens that emit light when struck by electrons. The light is recorded by video cameras and the images from those cameras are interpreted by a video frame grabber. The PAL hides the process of capturing beam characteristics from these devices and only exposes important features of the process, like conflicts in use of the frame grabber, or position information from the monitors.

The organization of controllers and solvers reflects an "all data is local" design. For this reason, information is only shared between controllers through global mechanisms, such as the plant model or the PAL, or through message passing between controllers. Message passing is the primary means for organizing control actions and distributing data throughout the system. Messaging typically occurs between parent and child controllers and is used to pass task information, convey system state, inform a parent of progress toward accomplishing some goal, or request assistance in satisfying a set of constraints.

Teleo-reactive Sequencing

Controllers in this architecture use an intelligent executive mechanism called teleo-reactive (TR) control (Nilsson 1994). Teleo-reactive behavior is intended to be both goal seeking and reactive. TR control thus occupies a region between conventional feedback-based control and discrete action planning.

TR programs sequence the execution of actions that have been assembled into a certain kind of goal-related plan. No assumption is made that the actions composing these plans are discrete or that an action's effects are completely deterministic. On the contrary teleo-actions are typically continuous and are executed as long as the action's preconditions hold

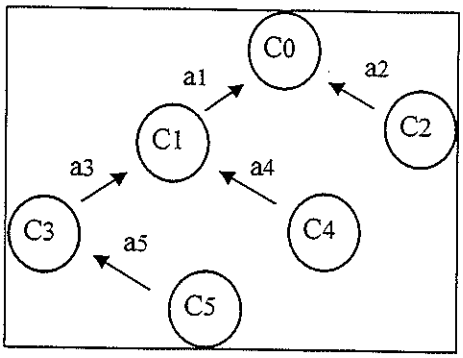

Figure 4. A teleo-reactive tree.

and its goal has not yet been achieved (unless some other action closer to the final goal becomes activated.) A short sense-react cycle ensures that when the environment changes, the control action changes to fit the new state.

A TR tree can be described as a set of condition-action pairs:

$$C0 \rightarrow A0$$
$$C1 \rightarrow A1$$
$$\ldots$$
$$Cn \rightarrow An$$

where *Cs* are conditions and *As* are the associated actions. *C0* is typically the top-level goal of the tree and *A0* is the null action, i.e., do nothing if the goal is achieved. At each execution cycle the *Ci*s are evaluated from top to bottom until the first true condition is found. The associated action *Ai* is then performed. The evaluation cycle is repeated at a relatively high frequency to simulate continuous reactivity.

TR action sequences or plans can also be represented in a data structure called a TR tree (Figure 4). TR trees are constructed so that each action *Ak*, if continuously executed under normal conditions, will eventually make *some* condition higher in the tree true. This ensures that under normal conditions the top-level goal, *C0*, will eventually become true.

TR tree execution is adaptive in that if some unanticipated event in the environment reverses the effects of previous actions, TR execution will fall back to a lower level condition and restart its work towards the top-level goal. On the other hand, when something "good" happens, TR execution is opportunistic. If a condition higher in the tree is suddenly satisfied, execution shifts to the action associated with that higher condition.

Teleo-reactive plans provide a very intuitive framework for encoding the control logic followed by accelerator operators. Teleo-reactive execution has been implemented in control systems for beamline tuning at Brookhaven and Argonne National Laboratories. It has been particularly effective in coordinating the alternation between linac tuning, steering and focusing operations, adaptively correcting steering each time a linac tuning or focusing operation induces incidental orbit changes.

The teleo-reactive mechanism provides a useful framework for representing and implementing accelerator tuning algorithms for several reasons:

- Accelerator beams and their associated diagnostics are typically dynamic and noisy.
- Achievement of tuning goals is often affected by stochastic processes such as RF breakdown or oscillations in the source.
- Many of the actions used for tuning are durative. This is often true of tweaking and optimization operations: they are typically done until specific criteria are met.
- TR trees provide an intuitive framework for encoding tuning plans acquired from accelerator physicists.

Argonne ATLAS physicist Richard Pardo, for example, rapidly encoded several teleo-reactive plans for tuning various segments of the PII beamline with relatively little outside assistance.

The accelerator control system described in this paper includes an online planner with the capacity to dynamically modify existing TR trees or construct new trees as a situation requires. Construction of TR trees is accomplished through a planning algorithm that is a modification of common AI goal reduction planning algorithms.

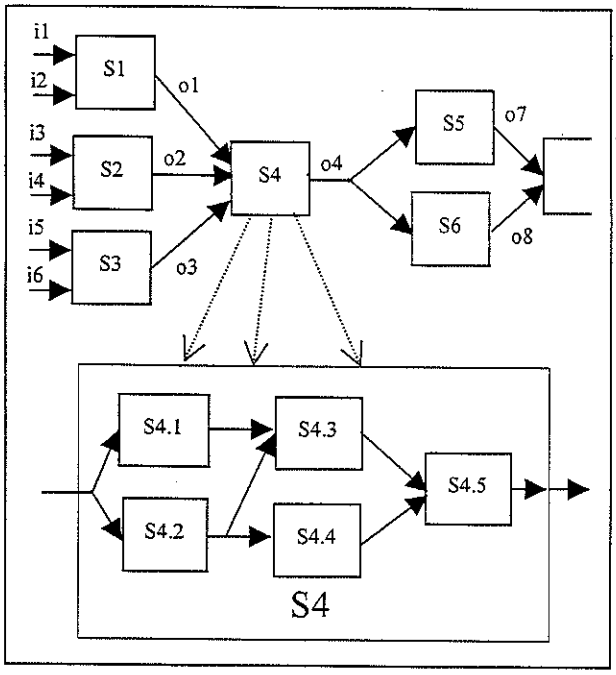

Figure 5. A hierarchical model representation.

Embedded Model-based Control

A major extension to this control framework is currently under development. The goal of this effort is to embed a model-based subsystem in a manner that is consistent with design philosophy of the control framework described above.

Models play a critical role in conventional accelerator control. However, they are generally used offline, first to derive initial beamline parameter settings, and later, as a framework for analyzing operational data.

Embedding models online will provide a number of important benefits. These include:

- The dynamic derivation of new parameter settings in response to new operational goals or subgoals.
- The potential for online recalibration of models when current model-derived parameter settings are significantly in error.
- The use of models for causal analysis and automatic diagnosis of hardware failures.
- The use of models to define normal operating conditions and constraints to be enforced by a process monitoring subsystem.

A framework has been developed for embedding online models that reflects the hierarchical component-based architecture described above. Models are defined hierarchically in terms of the connection topology between functional systems or components. A functional subsystem that is described at one level as a black box with a specific input-output function is decomposed at a lower level of the hierarchy into a connected assembly of components (Figure 5).

This model-based framework supports a principled and intuitive organization of the control system's graphical user interface. The hierarchy of displays through which operators view the functioning of the beamline corresponds to the hierarchical decomposition of the beamline model, with each display corresponding to a particular subsystem or component in the model. The displays are linked together to allow simple navigation to the most appropriate level of the hierarchy for addressing current operational objectives.

The use of a knowledge-based approach, particularly in the higher levels of the control hierarchy, necessitates two representations of the model. The accelerator model used by physicists to calculate operating parameters represents model components through continuous numerical scalars, vectors, and matrices. On the other hand, the perspective taken by knowledge-based controllers to reason about chains of causal influence and to determine whether components or subsystems are in normal or abnormal states involves discrete symbolic representations. Mediating these two types of representations is a mechanism implemented in the PAL, for discretizing continuous variables through translation into numerical ranges or fuzzy set variables. This mechanism is also used to translate symbolic expressions of invariants, constraints, and normality conditions into continuous numerical values and ranges.

The combination of these two views in the operator interface provides a powerful mechanism for both software and human monitoring. Using displays that combine abstract and concrete model-based views provides operators with a more intelligible framework for comparing the system's actual behavior with its expected or normal behavior.

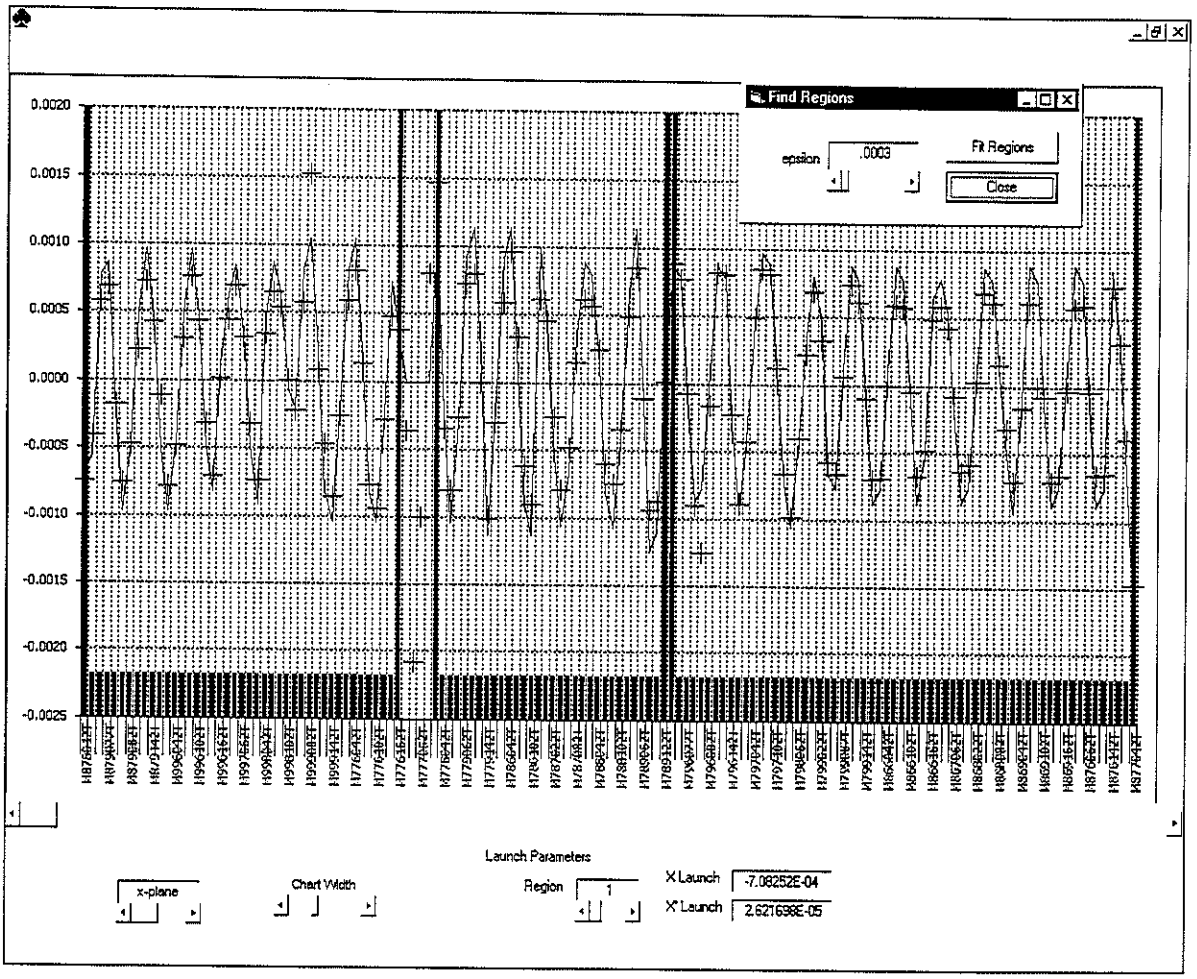

Figure 6. Finding good and bad regions in the high energy ring of PEPII.

Automating Error Correction in Accelerator Control Models

One of the advantages of embedding models online is the possibility of dynamically adapting or correcting models on the basis of experience in control operations. If a control operation using parameters derived from a model proves ineffective then that implies a disagreement between the model and the physical system. This disagreement can then be localized and traced backwards to its source by constructing a sequence of scenarios in which model-based predictions are compared with observations of system behavior. This is the approach used in a system that has been implemented for automatic error correction and recalibration of accelerator optics models [Stern & Luger 1999].

This diagnostic procedure relies on orbit deflection for probing the magnetic fields of focussing or *quadrupole* magnets. A set of steering or dipole magnets are tweaked, one by one, imparting a "kick" or deflection to the beam trajectory. After each kick, the beam's location at beam position monitors is measured. Each steering magnet thus generates a "track", i.e., an actual orbit that can be compared with the predicted orbit from the model.

A modeling code called *COMFORT* is used to calculate the expected orbit or trajectory of the beam through the beamline. *COMFORT* is fed an input file, called a "deck", with descriptions of every beamline component. Using *COMFORT* one can calculate the expected position of the beam after deflection at any desired location.

This first step in analysis of orbit response data is to apply a decompositional approach to separate the beamline into "good" and "bad" regions (Figure 6). Roughly speaking, good regions are regions where the model's predictions match the actual measurements up to some degree of tolerance. Bad regions are regions where the discrepancy between model and observation exceeds the specified tolerance.

A bit of cleverness is required in order for the method to work successfully. Without this technique, one would observe orbit errors propagating globally and be unable to identify local regions where observations and model-based predictions are in agreement. The position and angle of the beam on entry to the local region under analysis are initially treated as variables. The values of the entry position and angle variables are computed in the course of a least squares fit to the sequence of position observations at monitors in the local region. One enforces the condition that this least squares fit conform to the precise transformations in position and angle predicted by the model. These are encoded in transform or $\mathbf{R}$ matrices that are given by the model. It is then determined whether some initial position and angle allows successful fitting to the observed sequence of beam positions under the constraints imposed by the $\mathbf{R}$ matrices. If fitting is successful, the initial position and angle of the beam in this good region are called the "launch conditions".

Treating the entry conditions for a region as free variables that are resolved in the course of fitting allows for decoupling of local and global error. This is a general method invented at SLAC in the 1980s that is of critical importance in model-based analysis (Lee & Clearwater 1987). *Global* error is captured and absorbed into the entry conditions for a *local* region, freeing the local region of global error and allowing a determination as to whether model-based prediction and observation agree locally.

Once good and bad regions are resolved, the next step is to analyze and correct the errors in the bad regions. The fact that orbit errors become observable within a short distance of the components that produce them insures that all errors occur within the neighborhood of bad regions, and conversely that good regions contain only minimal discrepancies between prediction and observation.

Bad regions are analyzed using a combination of numerical fitting and optimizing search. First, the observed sequence of beam positions over multiple tracks are collected into a system of linear equations from which $\mathbf{R}$ matrices for the local region are derived. These $\mathbf{R}$ matrices, based on BPM data, are the transform matrices for the actual machine. Of course, in bad regions these actual $\mathbf{R}$ matrices do not match those in the model. Thus, the next step is to determine what changes in the model are required to bring the model's predictions into agreement with the observations.

COMFORT, like most accelerator modeling codes, provides fitting services that vary the strengths of magnets in the model to approximate a specified $\mathbf{R}$ matrix for a given region. To use this fitting service, one specifies a set of magnet field strengths to vary and a set of target parameters to fit. Each *COMFORT* fitting returns a measure of the degree to which the fitting was successful.

The quality of the fitting is determined by the degree to which some set of changes in the field strengths of a particular set of magnets will reproduce the observed orbit in the bad region. These relationships allow one to use a sequence of fittings to search for the actual set of miscalibrated magnets in the bad region. This fitting procedure also gives a new set of magnet field strengths that brings the revised model into agreement with observed system behavior.

Results in Accelerator Diagnosis and Control

The accelerator control system that is described here has been tested on the Brookhaven ATF beamline and the Argonne ATLAS beamline. The portable nature of the design was demonstrated by moving the control system from Brookhaven to Argonne in a six week time period with less than three programmer weeks of effort (Klein et al. 1997).

Tests of automated tuning operations also proved successful. The control system was used to implement automated tuning operations at both Brookhaven and Argonne. The automated tuning system repeatedly achieved tunes equal in quality to the best achieved

by human operators, typically in less time (Klein et al. 1997).

The automated system for error correction and recalibration of optics models has been successfully applied to identify errors in the high-energy ring (HER) of PEPII. It identified two errors, one of which has also been confirmed by manual analysis (Stern 1999). It has performed well on simulated data from the SPEAR ring and will soon be exercised in a double blind test on real data from SPEAR to see if it can identify quadrupole magnets whose field strengths have been deliberately altered.

Summary

An intelligent hierarchical component-based framework integrating model-based diagnosis and control is described. This framework has provided the basis for development of a diagnostic and control system for particle beam accelerators. Control automation results achieved in the accelerator domain are quite promising, in many respects exceeding those achieved by previous efforts. A system integrating automatic error analysis and recalibration of control models has also been implemented and successfully tested, breaking new ground in the domain of accelerator control.

The framework described here was intended to provide a general framework for the control of large complex systems. It is expected that these design concepts will prove applicable in the development of systems for the control of large integrated systems other than particle accelerators, including possibly those on naval ships. This of course will require further analysis.

References

Clout, P. 1993. The status of Vsystem. *Proceedings of the Third International Conference on Accelerator and Large Experiment Physics Control Systems*, Germany.

Klein, W., Stern, C., Luger, G., and Olsson, E. 1997. An Intelligent Control Architecture for Accelerator Beamline Tuning. *Proceedings of the Innovative Applications of Artificial Intelligence Conference*, 1997. Cambridge MA: MIT Press.

Lee, M., and Clearwater, S. 1987. Generic orbit and lattice debugger. *Proceedings of the Model Based Accelerator Controls Workshop*, Brookhaven National Laboratory.

Nilsson, N. J. 1994. Teleo-Reactive Programs for Agent Control. *Journal of Artificial Intelligence Research*, 1, pp. 139-158, January 1994.

Acknowledgements

We wish to thank the Department of Energy for a series of SBIR grants, DE-FG05-95ER81897 and DE-FG03-998ER82708/A000, supporting research on intelligent accelerator control. The National Science Foundation is also to be thanked for its support of research in the diagnostic applications of probabilistic causal networks (Bayesian networks) at the University of New Mexico. The far-sighted support of research in intelligent control automation by Vista Control Systems, Inc., and particularly Vista's president, Peter Clout, has made this work possible.

Carl R. Stern, Ph.D., *is the principal author and the President of SandiaView Software, Inc. He is the former Manager of Research at Vista Control Systems, Inc. His degree is in Computer Science with a specialization in Artificial Intelligence. He is also an NSF funded Research Associate in the Department of Computer Science at the University of New Mexico. His specialization is in diagnostic technologies and in the integration of model-based diagnosis and control.*

George F. Luger, Ph.D, *is a Professor of Computer Science at the University of New Mexico. His research area is Artificial Intelligence. He is PI on an NSF grant with Carl Stern on the application of Bayesian networks to diagnostic problem solving. He is co-author of a widely used textbook on artificial intelligence now in its third edition:* ***Artificial Intelligence: Structures and Strategies for Complex Problem Solving****, London: Addison-Wesley, 1998. He also serves on the Board of AAAI Press.*