AUTOMATED DIAGNOSIS OF PHYSICAL SYSTEMS

S. Narasimhan, UC Santa Cruz @ NASA Ames Research Center, CA, U.S.A.

Abstract

"If anything can go wrong, it will." (Murphy's Law) Although deceptive in its simplicity, the above quote has proved to be a profound insight into how things happen. Things are likely to fail no matter how well the system is designed. When a high degree of reliability and safety is desired the effects of these failures must be mitigated, and control must be maintained under all fault scenarios. Faults need to be detected close to their onset so that quick action can be taken by resetting control parameters to compensate for the fault or by reconfiguring the system to minimize the effects of the fault and thus prevent damage. In this paper we provide a brief introduction to variety of automated techniques for diagnosing faults and then discuss in more detail one specific technology called HyDE.

WHAT IS AUTOMATED DIAGNOSIS?

Fault diagnosis involves the detection of anomalous system behavior and the identification of the cause for the deviant behavior. *Automated* diagnosis refers to the use of software technologies to assist in diagnosing faults in a system. This is to be contrasted with *autonomous* diagnosis which deals with software technologies that operate autonomously to detect, isolate and compensate for faults in a system.

We would like to introduce some definitions (taken from IFAC Technical Committee SAFEPROCESS [1]) to set the context for the rest of the paper.

Fault - A fault is an unpermitted deviation of at least one characteristic property or parameter of the system from acceptable, usual or standard conditions.

Fault Detection - Fault detection is monitoring measured variables to determine if a fault has occurred in the plant. If a fault has occurred, it may be important to determine the time at which the fault occurred.

Fault Isolation - Fault Isolation is determining the type and location of a fault once it is known that a fault has occurred. It typically follows fault detection but the two processes are often combined for additive faults.

Fault Identification - Fault Identification is determining the size and time-variant behavior of a fault. It follows fault isolation.

Fault may be of many types including:

- Plant, actuator, sensor or controller faults
- Additive or multiplicative faults
- Abrupt or incipient faults
- Persistent or intermittent faults

Diagnosis is made harder by several factors including and not limited to:

- Typical only a few sensors are placed leading to limited observability into system behavior.
- The data from sensors are noisy due to inherent properties of the sensors
- There are unknown inputs acting on the systems, due to lack of complete knowledge about conditions in which system operates.
- The knowledge about how the system functions may be limited.
- The effects of faults may be non-local and may take some time to manifest.

In the rest of this paper we will introduce a sampling of automated diagnosis techniques that deal with some of the fault characteristics described above. We will briefly describe these techniques and then focus on one specific technique called HyDE.

AUTOMATED DIAGNOSIS TECHNIQUES

The diagnosis problem has been dealt with in several domains from several angles resulting in a wide variety of diagnosis approaches. Some of these include Expert Systems, Case-based reasoning, Data-driven techniques and Model-based reasoning.

Expert System Diagnosis [2,3]

Traditionally diagnosis was performed by human troubleshooting experts who built up diagnostic knowledge based on their expertise and experience. The natural extension to this was to encode the diagnostic knowledge in a machine storable structure. These structures took the form of associations between observed symptoms and probable fault occurrences. Tools were built to assist in the creation of these structures and then to use these structures for the diagnosis task. Once these structures are built, users and non-expert operators could troubleshoot the system in case of faults.

Some of the more commonly used structures to encode expert diagnostic knowledge are *rules* and *fault trees*. A rule describes the action(s) that should be taken if a symptom is observed. A set of rules describing the symptoms of all the possible faults is incorporated into a rule-based reasoning system. The reasoning may use a backward-chaining algorithm which starts at the hypothesis (consequents of rules) and collects and verifies evidence (antecedents of rules) that supports the hypothesis. Alternately forward-chaining may be used where rules whose antecedents match observed symptoms are examined. When several rules match, a chain of rule firings (based on pre-defined rule priority) is used to establish the diagnosis. A fault tree (decision tree) encodes diagnostic knowledge as a sequence of questions that trace a path from the root of the tree to its leaf nodes which represent diagnoses.

Advantages of expert systems are:

- Diagnosis structure is "certified" by experts and can be trusted to produce "correct" results.

- In many cases, deep understanding of the physical properties of the system is either unavailable or too costly to obtain.
- The diagnostic reasoning is fast and bounded both temporally and computationally.

Disadvantages of expert systems are:

- All fault-symptom manifestations have to be encoded for "correct" diagnosis.
- Troubleshooting experts may not be available. Even when they are available it takes years of experience for them to gather all the diagnostic knowledge.
- The whole process has to be started from scratch for each new application.

Case-based Diagnosis [4,5]

Case-based Reasoning Systems exploit knowledge about solutions developed for past problems to solve current problems. In this approach, experiences are stored in the form of diagnostic cases. When a new case needs to be diagnosed, stored cases are scanned to find any matches with the new case. The new case is then added to the library of stored cases under an appropriate category.

Like rule-based systems, past experience with normal and abnormal behavior of a system are essential to building effective case-based diagnosis systems. In addition, case-based reasoning systems include a learning component which makes possible adaptation of a past solution to fit other, similar situations. This technique is well suited for poorly understood problem areas for which structured data are available to characterize operating scenarios.

A case-based reasoning system consists of a case library containing features that describe the problem, outcomes, solutions, methods used and an assessment of their efficacy. A coding mechanism is used to index the case information so that the cases can be organized into meaningful structures, such as clusters, enabling efficient retrieval.

Advantages of case-based reasoners are:

- Only experience in the form of past solutions is needed rather than a deep understanding of the physical properties of the system.
- Diagnosis of cases that have been seen before is very fast.
- It can be directly applied to any new application.

Disadvantages of case-based reasoners are:

- New "cases" cannot be diagnosed.
- A lot of prior experience is needed to build a large database of cases.

Data-driven Diagnosis [6-10]

- Data-driven approaches are based on the assumption that statistical characteristics of monitored data from the system indicate abnormal events in the system. These techniques transform the high-dimensional noisy data into lower-dimensional information for detection and diagnostic decisions allowing the ability to handle highly collinear data of high dimensionality, substantially reduce the dimensionality of the monitoring problem, and compress the data for archiving purposes.
- Diagnosis using these approaches typically involves two steps:
  1. Learning Step – Prior to diagnosis, data from various operational scenarios of the system is fed to a learning algorithm which computes characteristics of the data sets (in a much lower-dimensional form). A classifier then tries to classify the lower dimensional characteristics into groups with similar properties. The classification may be based on prior knowledge about the actual operational scenarios.
  2. Diagnosis Step – The learning algorithm is applied on the data from real-time operation of the system to compute lower-dimensional characteristics. These characteristics are compared against learned groups (from the classifier) to select the closest group as the diagnosis.

Some of the approaches that use data-driven techniques are Regression analysis, Principal components analysis, Artificial neural networks, Filters, Harmonic analyzers, Auto and cross-correlation functions, Fast Fourier transform (FFT), Pattern Recognition, Feature Selection, Model Selection, Ensemble Learning, and Support Vector Machines.

Advantages of data-driven diagnosis are:

- No understanding of system properties is necessary. Only data from operation of system is necessary.
- Very high dimensional and noisy data can be handled without any problems.
- The diagnostic inference is bounded and can be very fast.
- Generic algorithms can be applied on any data set from any application.

Disadvantages of data-driven diagnosis are:

- A lot of data about various diagnostic scenarios is necessary for proper classification.
- The diagnosis results are very sensitive to the data used.

Model-based Diagnosis [11-22]

Model-Based Diagnosis (MBD) departs from these approaches by using a model of the system configuration and behavior. MBD exploits the analytical redundancy (functional relationships) between the model and the system measurements. In principle, this means that a model runs in parallel to the real process, and discrepancies between the model outputs and real outputs are utilized to detect and isolate faults. Some of the major categories of model-based diagnosis techniques are consistency-based approaches, control-theory based approaches, and stochastic approaches.

Consistency-based Diagnosis [11-15]

In this approach an abstract model of the system is used for diagnosis. The model may be discrete, discrete-event, or continuous in form and typically includes only information necessary to diagnose faults. The faults are represented as changes in operational modes of components of the system. The model is used to predict the expected behavior of the system under hypothesized conditions for operational modes of the system. The predictions are compared against observations to determine discrepancies. Discrepancies are used to generate conflicts which drive a search process for alternate hypothesis for operation modes. The hypothesized operation modes that predict behavior closest to observations are reported as diagnosis. Some of the technologies that use this approach are GDE/Sherlock, TRANSCEND, and Livingstone/L2.

Control-theory based approaches [16-18]

Quantitative diagnostic algorithms derived from the control theory community use state estimation and parameter estimation techniques for diagnosis. A mathematical model of the system is used to perform diagnosis as a two step process:

1. Residual Generation - Residuals are generated from the model and information about inputs and outputs of the system. Residuals are variables with zero value when the plant operation is nominal and non-zero otherwise, and
2. Decision Making - Decision procedure is applied to discriminate non-zero residuals that are result of modeling errors, unknown inputs, measurement errors and those which reflect abnormal behavior.

The residual generator enhances the raw residuals in one of the following two ways:

1. In response to a fault, only a specific set of residual vector elements become non-zero (structured residuals), or
2. In response to a fault, the residual vector lies in a specific direction (fixed directional residuals).

The decision maker takes these enhanced residuals and evaluates them to determine the fault. In the case of structured residuals, this involves mapping the specific set of non-zero residuals to faults. Some of the approaches that use this method are structured parity equations, structured residuals from state equations, diagnostic observer design with direct eigen-structure assignment, and unknown input observer in Kronecker canonical form. In the case of the fixed directional residuals, the decision maker finds pre-defined fault residual direction that is closest the direction of the actual residuals. Some of the approaches that use this method are detection filter design by eigen-structure assignment, fixed direction residuals from parity equations, and matched filters.

Stochastic Approaches [19-21]

In recent years there has been a push towards the use of probabilistic approaches for diagnosis. This is motivated by the fact that a lot of uncertainty exists in the diagnosis process. The sources of uncertainty may be the models, the sensors, the environment, the diagnosis algorithms etc. These techniques maintain a belief state (a set of possible diagnoses ranked by probabilities) about the system and update the beliefs using probabilistic update mechanisms. Some examples include Bayes Nets, Kalman Filters and Particle Filters.

Advantages of model-based diagnosis are:

- Model libraries can be re-used.
- Reasoning algorithms remain the same for all applications.

Disadvantages of model-based diagnosis are:

- Someone has to build models.
- Reasoning time is not bounded and my take very long.

HYDE

We will now talk about one specific model-based diagnosis technology that is attempting to integrate some of the techniques describe above. HyDE (Hybrid Diagnosis Engine) [22] combines ideas from consistency-based, control-theory-based and stochastic approaches to provide a general, flexible and extensible architecture for stochastic and hybrid diagnosis. HyDE supports the use of multiple paradigms and is extensible to support new paradigms. HyDE also offers a library of algorithms to be used in the various steps of the diagnostic reasoning process. The key features of HyDE are:

- Diagnosis of multiple discrete faults.
- Support for hybrid models, including autonomous and commanded discrete switching.
- Support for stochastic models and stochastic reasoning.
- Capability for handling time delay in the propagation of fault effects.

HyDE Models

HyDE models have two parts, the transition model and the behavior model. The transition model describes the components that make up the system, the various operating modes of the system (include faulty ones), and the conditions for transitions between the operating modes. The behavior model specifies the behavior evolution and has three parts: propagation model, integration model and dependency model. The information in the propagation model allows the estimation of unknown variable values from known variable values. The dependency model captures information about the dependencies between variables, models and components. The integration model describes how the variables' values are propagated across time steps. HyDE supports the representation of each of the behavior models in more than one paradigm.

HyDE Reasoning

HyDE reasoning is the maintenance of a set of weighted candidates. A candidate represents the hypothesized trajectory of the system inferred from the transition and behavior models, knowledge of the initial operating modes of all components and initial values of all variables, and the sensor observations reported to HyDE. The candidates' weights are a way of ranking them and depend on several factors, including prior probabilities of transitions and the degree of fit between model predictions and observations. Although weights are in the range [0, 1], weight is not a probability measure, specifically posterior probability.

Each candidate contains a possible trajectory of system behavior evolution represented in the form of a hybrid state history and transition history. The hybrid state is a snapshot of the entire system state at any single instant. It associates all components with their current operating modes and all variables with their current values. Applications run HyDE at discrete time steps, typically but not necessarily when observations are available. Time steps need not be periodic. For each time step that HyDE reasons about, a candidate contains two hybrid states, one at the beginning of the time step and one at the end, as well as the set of transitions taken by the system between the previous and current time steps.

At time step 0 the candidate set is initialized with candidate(s) derived from the initial hybrid state of the system. Once the initial candidate set has been created, HyDE's reasoning process uses the same sequence of operations for each time step. The reasoning process can be divided into three categories of operations,

1. Candidate Set Management maintains the candidate set. The operations include updating weight of all candidates, pruning candidates that do not satisfy minimum weight requirements, adding new candidates (the next best ones from the candidate generator) when necessary, and optionally re-sampling or normalizing the distribution of weights.
2. Candidate Testing deals with operations on a single candidate. The operations include determining the occurrence of any transitions, estimating the hybrid states at the beginning and end of a time step, comparing against observations to update weight of the candidate as well as reporting inconsistencies.
3. Candidate Generation creates candidate generators from inconsistencies reported by Candidate Testing and supplies the next-best potential (untested) candidate to Candidate Set Management when requested. This is achieved using a conflict directed search. First reported inconsistencies are used to generate conflicts (subset of operating modes that cannot all be true at the same time). The conflicts are then used to guide a search for new candidates optimizing some candidate property (typically weight or size).

HyDE Implementation Status

The HyDE reasoning engine is implemented in C++. Complete diagnosis reasoning can be performed. It passes an extensive and demanding test suite on Windows, Solaris, Linux and VxWorks platforms. A graphical modeling environment is available using the GME open-source tool [23]. The same environment can also be used to set initial state and configuration parameters. Observations can be reported to HyDE either through streams (file or otherwise) or an API allowing integration with a real-time system. HyDE binaries are available free of cost for non-commercial purposes.

HyDE Applications

HyDE has successfully used (and continues to be used) on several projects including:

- The Drilling Automation for Mars Environment (DAME) project is aimed at developing a lightweight, low-power drill prototype that can be mounted on a Mars Lander and drill several meters below the Mars surface for conducting geology and astrobiology research. Three kinds of diagnosis technologies were used on this project, HyDE for model-based diagnosis, a rule-based diagnosis system, and a neural-network diagnosis system. There were four rounds of testing over a period of two years. In 2005, laboratory experiments were run at Honeybee Robotics, the company that constructed the drill. Later in 2005, field tests were performed at Houghton Crater on Devon Island, in the Canadian arctic, chosen to approximate the Martian terrain and climate. Based on the results, laboratory tests were performed at NASA Ames Research Center in 2006 to improve the models for better diagnosis. Finally, there was a second field test at the Houghton Crater. A summary of HyDE's final performance: All of the modeled fault modes were encountered in the field; some, such as choking, binding, and hard material, numerous times. The model-based diagnostic system was able to successfully identify the faults in roughly 85% of the cases. The rate of false positive diagnoses was approximately 5%.
- The Advanced Diagnostic and Prognostic Test-bed (ADAPT) was developed to test, measure, evaluate, and mature diagnostic and prognostic health-management technologies. The initial test-bed configuration is functionally representative of an exploration vehicle's Electrical Power System. HyDE was used to build an 86-component model of the test-bed components. HyDE passed all the acceptance tests, which included a set of pre-defined fault scenarios and acceptance bounds on the time to diagnosis. Additional tests were run on other fault scenarios (not included in acceptance tests) and HyDE was able to diagnose all but one of these scenarios (resulting from inadequate information in the model).
- The Autonomous Lander Demonstrator project (ALDER) demonstrated autonomy capabilities relevant to a small spacecraft mission on traditional flight hardware integrated with traditional flight software. Diagnosis systems, including HyDE, were ported to VxWorks and executed on the Aitech S950 processor, interoperating with autonomy technologies for planning, diagnosis, computer vision and adaptive control. A HyDE component model of a simple propulsion system (tanks, valves, regulators, attitude control system and main engine) detected common faults such as stuck valves and regulator failures.
- Several other projects are using HyDE for offline diagnosis using simulated data. The International

Space Station (ISS) Electrical Power System (EPS) recovery procedure automation project uses HyDE to supply current hybrid state to an execution system that will attempt to partially automate recovery procedures based on this information. The Aircraft Landing Gear Diagnosis project is using HyDE to model the landing gear and wheels of an aircraft in an effort to diagnose faults during landing. The Spacecraft Engine Diagnosis project uses HyDE to model components of the J2X engine, which is expected to power the upper stage of the NASA Crew Launch Vehicle (CLV). HyDE was also integrated with the CLARAty (Coupled Layer Architecture for Robotic Autonomy) architecture.

HyDE Future Work

HyDE is still a work in progress and we expect to keep adding to the capabilities to HyDE through addition of new modeling paradigms and algorithms. We have identified several major areas for future work. We would like provide mechanisms for validation and verification (V&V) of models and algorithms, which would be prerequisite for deployment on real systems. We would like to add support for parametric faults. This would require support for modeling of parameters representing such faults, additional algorithms for parameter estimation and modifications to existing algorithms for detection and isolation of such faults. We would also like to use the existing models and algorithms for suggesting recovery sequences.

References

[1] R.J. Patton and J. Chen, "Robust Model-based Fault Diagnosis for Dynamic Systems" Boston, 1999, Kluwer Academic Publishers.

[2] Giarratano, Joseph C. and Riley, Gary D., "Expert Systems: Principles and Programming", Fourth Edition, PWS Publishing Company, Boston MA, 2004.

[3] Buchanan, B.G. and Shortliffe, E.H., editors, "Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project", Addison-Wesley, 1984.

[4] Aamodt, A., Plaza, E.: "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", AI- Communications, 7 (i), pp. 39-59. 1993.

[5] Varma, A. and Roddy, N., "ICARUS: A Case-Based System for Locomotive Diagnostics," Engineering Applications of Artificial Intelligence Journal, 1999.

[6] Duda, R.O., Hart, P.E., and Stork, D., Pattern classification, John Wiley & Sons, New York, 2000.

[7] Bishop, C.M., Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1997.

[8] Cherkassky, V., and Mulier, F., Learning from data, concepts, theory and methods, John Wiley & Sons, New York, 1998.

[9] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J., Classification and Regression Trees, Wadsworth, California, 1984.

[10] Quinlan, J.R., C4.5: Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann, 1993.

[11] W. Hamscher, L. Console, and J. De Kleer. "Readings in Model-based Diagnosis." San Mateo, CA: Morgan Kaufmann, 1992.

[12] J. De Kleer and B. C. Williams. "Diagnosing multiple faults", Artificial Intelligence, 32(1):97–130, 1987.

[13] P. J. Mosterman and G. Biswas. "Diagnosis of continuous valued systems in transient operating regions". IEEE Transactions on Systems, Man, and Cybernetics, 1(6):554–565, 1999.

[14] B. Williams and P. Nayak. "A model-based approach to reactive self-configuring systems", in AAAI, pp. 971–978, (1996).

[15] J. Kurien and P. Nayak. Back to the Future with Consistency-based Trajectory Tracking, AAA/IAAI 2000, pp370-377

[16] White, J.E. and J.L. Speyer. Detection Filter Design: Spectral Theory and Algorithms. IEEE Transactions on Automatic Control, 1987. 32: pp. 593-603.

[17] Chow, E.Y. and A.S. Willsky. Analytical Redundancy and the Design of Robust Failure Detection Systems. IEEE Transactions on Automatic Control, 1984. AC-29: pp. 603-614.

[18] J. Gertler. "Fault Detection and Diagnosis in Engineering Systems". New York: Marcel Dekker, 1988.

[19] M. Hofbaur and B. Williams. "Mode Estimation of Probabilistic Hybrid Systems", in Proc. 5th International Workshop on Hybrid Systems: Computation and Control (HSCC '02), Stanford, CA, USA, pp. 253-266, 2002.

[20] R. Dearden and D, Clancy. "Particle Filters for Real-time Fault Detection in Planetary Rovers", in Proc. 13th International Workshop on Principles of Diagnosis (DX '02), Semmering, Austria, pp. 1-6, 2002.

[21] S. Narasimhan and G. Biswas. "Model-based Diagnosis of Hybrid Systems". Eighteenth Intl. Joint Conf. on Artificial Intelligence, Acapulco, Mexico, Aug., 2003.

[22] S. Narasimhan and L. Brownston. "HyDE – A General Framework for Stochastic and Hybrid Model-based Diagnosis", in Proc. 18th International Workshop on Principles of Diagnosis (DX '07), Nashville, USA, pp. 162-169, 2007.

[23] Ledeczi A., Maroti M., Bakay A., Karsai G., Garrett J., Thomason IV C., Nordstrom G., Sprinkle J., Volgyesi P.: The Generic Modeling Environment, Workshop on Intelligent Signal Processing, Budapest, Hungary, May 17, 2001.