The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Adding diagnostics to intelligent robot systems

Sneha Chandrababu and Henrik I. Christensen

Abstract— **Robot systems are increasing in complexity. Trying to diagnose a robot that is non-functional or exhibiting sub-optimal performance can be a major challenge. A framework for plug-n-play addition of diagnostics to modules in an object oriented software framework is presented. The methods for modeling of system modules, their transition to a Bayesian model and final implementation are described. The methodology is exemplified for a mobile manipulation system and experimental results are presented.**

Keywords: Diagnostics, Intelligent Robotics, Bayesian network, Troubleshooting, Quantitative Modeling

I. Introduction

For anything but a trivial application there are numerous hardware and software modules involved in the implementation of a system. As complexity grows, so does the difficulty of understanding the system and its behavior. A tedious aspect of robotics is that it can be time consuming to work with a real system due to the challenges of real-world experiments. Often a "relatively" simple problem can be very hard to diagnose as the over-all behavior of the system interacts with the defective device/module in a non-trivial way. Consequently diagnostics of a system is an open challenge.

In other fields there have been significant progress on design of methods for diagnostics [1], [2]. As an example all printers today have mature methods for error recovery and diagnostics [3], and the same is true for modern cars [4], hydraulic forging presses [5], gas turbines [6] and electronic systems [7]. For some reason relatively little work has been performed on transition of these methods to robotics.

Bayesian network models provide the basic mechanism for doing statistical diagnostics. The Bayesian network model creation is non-trivial and might involve long discussions with domain experts and technicians, and extensive research of the system under consideration. It involves a qualitative modeling part during which we elicit the structure of the Bayesian network. In the quantitative modeling part we elicit the statistical information, including conditional probability distributions and utilities [8].

In this paper a systematic methodology for modeling of errors in a robot system is described. Earlier work is presented in Section II and the overall design aspects are presented in Section III. The framework is exemplified for a system to make the effort concrete. The baseline system is presented in Section III-A and the qualitative design of diagnostics models is discussed in Section III-B. The augmentation of the system to have diagnostics capabilities is presented in Section IV. Example performance is presented in Section V and a summary is available in Section VI.

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, USA {sneha.chandrababu,hic}@cc.gatech.edu

II. Related Work

The area of diagnostics and use of expert systems dates back more than 50 years and tremendous progress has been achieved. There are now basically four different paradigms that are used for this task. They are rule based, case based, decision tree based, and model based expert systems each of which have their advantages and disadvantages. Of these, the model based expert system deals with probabilistic networks that are compact, intuitive representations of causal relations among the entities of the problem domain [8].

The other three models, though they have their advantages, are less suitable for modeling complex, dynamic systems like robots. The rule based system [9] would result in a large collection of rules that could become exponentially difficult to maintain, while the case based system [10] is based on experiences and are focused on known errors. The decision tree based diagnostics [11] involves traversing through various tests until a fault is reached and hence provides limited flexibility.

Bayesian networks have emerged as the modeling technique of choice in various applications of diagnostics. Skaanning et al [12] discusses a real world application of Bayesian networks for printer systems. It provides valuable insights into developing a model for diagnosis and acquiring knowledge. They use a custom built system for troubleshooting printer systems. In Przytula [13] a procedure for efficient creation of Bayesian networks has been designed, which has been applied in diesel locomotives, satellite communication systems and satellite testing equipment [14].

The most wide spread application of Bayesian inference technique is the Microsoft Office Assistant [15] developed in the Decision Theory and Adaptive Systems Group at Microsoft Research. In the office assistant, observations are entered continuously into the model and the the user needs are inferred. In our approach to diagnostics we employ a similar technique of updating the network continuously with new observations to capture the system error state.

Various tools like Hugin [16], [17] and MSBNx [18] are available for generation of Bayesian network. In our implementation we use GeNIe [19] for modeling purposes and SMILE [20] for construction of a customized troubleshooter. These tools accelerate the building of Bayesian networks and creation of troubleshooters that could be customized for the application using API's provided in various languages like C++, Java and C#.

III. System Design

In this section we discuss the intelligent system that is used in the diagnostic model design. In section III-A, we give an overview of the system under study and the method

Fig. 1. Mobile manipulator

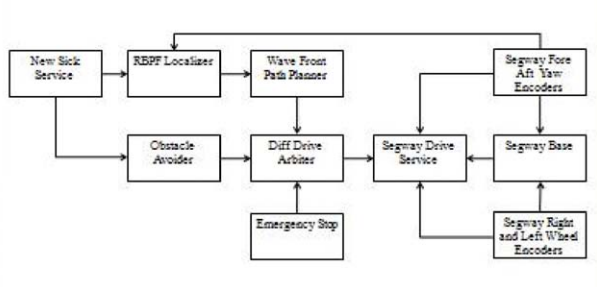

Fig. 2. Navigation system

for collection of information relevant to the construction of a diagnostics model. In section III-B we go into the details of the construction of a diagnostic model for the system studied.

In sub-section III-B.1 we discuss how the system modeling can be broken down into several phases and modules to enable an organized step by step modeling of the diagnostic system. In sub-section III-B.2 we describe the object oriented approach to developing the network that further reduces the complexity of building the network. Finally in sub-section III-B.3 we describe a distributed approach for diagnostics in an intelligent system, that would reduce the burden of information collection and processing for a central diagnostic network.

A. Intelligent System definition and study

The intelligent system under consideration for diagnostics is a mobile manipulation system. The hardware platform is composed of a KUKA light weight arm mounted on a Segway RMP200 base as seen in Fig. 1. The end effecter is a Schunk PG-70 2-finger parallel gripper. A Unibrain Fire-I firewire camera is mounted on the end effecter for visual servoing. The platform is equipped with a SICK LMS291 for localization and navigation. The software system is implemented using Microsoft Robotics Studio 1.5 [21]. The services that are implemented using MRS 1.5 can be used to represent anything from hardware components such as sensors and actuators to software components like UI, storage services and aggregation of sensors.

We took a subset of this system which consisted of the services belonging to the navigation module of the system and built a model of the diagnostic system for this sub-system seen in Fig. 2. The initial implementation of the system was made available to this project. The system is described in [22]. The system presented here is an augmentation to the originally implemented robot system.

The NewSick service represents the SICK Laser Range Finder and it distributes Laser scan information to those services that have subscribed to it for the laser scan packets. The Obstacle Avoider receives these messages and determines the direction and speed of travel. The RBPFLocalizer receives the laser scan packets and SegwayForeAftYaw encoder information and determines the current position and direction of the robot in the map. The WaveFrontPathPlanner uses the localization information to plan a path to the goal. The EmergencyStop service allows the user to issue a soft Stop signal to the robot. The DifferentialDriveArbiter service uses the inputs of the WaveFrontPathPlanner, ObstacleAvoider and the EmergencyStop services for the low-level turn and speed control reference. The SegwayDrive service maintains the drive states and implements the low level drive control for the platform.

The behaviour of each module changes over the life cycle of a process. A sample life cycle for the Segway Base service is shown in Fig. 3. The information for the model construction is elicited from the life cycle modeling of these services and the various documentations of the hardware used by the system. It is important that we include all the information relevant to the diagnostics of the system. The granularity of the model developed is set to a level at which a user of the diagnostic tool would be able to access and troubleshoot the system involving both physical modification of the hardware of the system and modifying states of the software services of the system. These involve normal troubleshooting steps like changing the parts of the system (e.g. SICK LRF), checking the power in a device (on / off / battery charge required) and modifying the serial port number in the state of services.

B. Diagnostic Model Design

1) System Breakdown: We adopt a high level view of the life cycle of all these services and modeling each phase using a Bayesian network. This high level view is shown in the Fig. 4 and is composed of four phases. The navigation module life cycle starts with the initialize phase during which a new instance of each of the Service classes is created. The services are tied to the network and assigned a URI. If partner services have been specified for the services they are started or found. Then the `start()` method of the services is activated.

The second phase is runtime during which the services run within a context known as the Decentralized Software Services (DSS) node. They can provide services and get notification from other services they are subscribed to and call methods of their partners to perform specific functions. When the system has completed its task it can go to the shutdown phase during which the service states are reset and the services are dropped. If anything goes wrong in any of these phases, the system moves to the Safe state during which the error in the system is identified and trouble-shooting is done to identify the cause

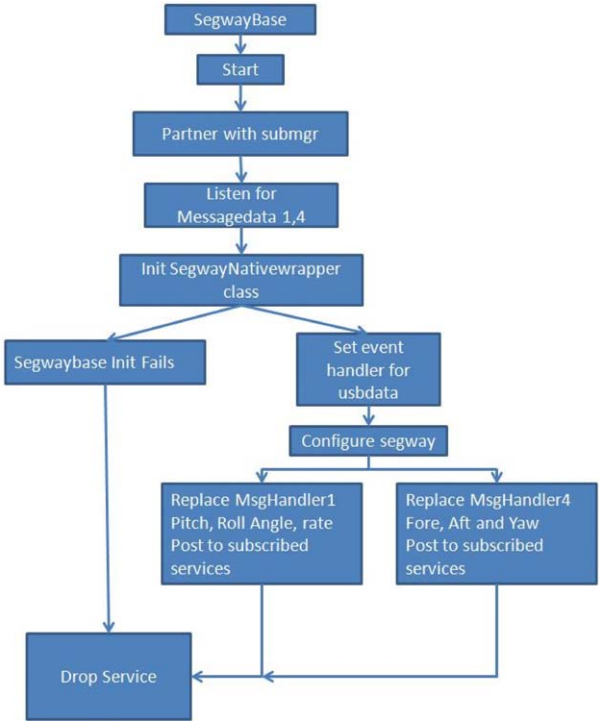

Fig. 3. Life cycle of segway base service

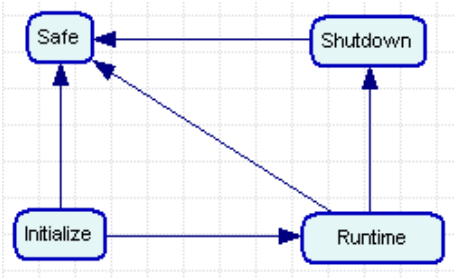

Fig. 4. Life cycle of services

of an error. In the safe state, the other functions of the system are brought to a stop (for example issuing an Emergency soft stop to the Segway).

Each of these phases of the navigation system runs smoothly depending on each of the component services initialization, runtime and shutdown phases.

2) Object Oriented Bayesian Networks: Each of these sub-systems might share several common features. For example, for all these sub-systems / services, the partnering of services is a common feature for initialization of the service. This introduces another way for simplifying the design of the diagnostic model of the system. These identical fragments of networks also share common conditional probabilities. When each of these services is modeled, the common causal networks are identified and place holder nodes (input /output nodes) are used to represent these features. The modeling of the sub-network is performed independent of the parent network. Thus only the input/ output nodes which are connected to the parent node remain the same. The rest of the hidden attributes of the network can be modified as required by the diagnostic model developer.

These sub-networks can be called classes and their instances are used within the bigger networks. These networks are made up of input nodes, output nodes and several encapsulated nodes. The encapsulated attributes are representative of the object oriented paradigm. By abstracting away these nodes/ attributes we obtain the Object Oriented Bayesian Network (OOBN) which gives a simplified view of the diagnostic model. It also avoids repetitiveness and thus makes use of the principle of re-usability of networks. When the OOBN is used for querying, the OOBN can be transformed to a complex Bayesian Network by merging each input/output attribute with the place holder nodes in the parent network. Thus the OOBN is expanded into a standard Bayesian network for inference purposes [23].

3) Distributed Model for Diagnostics: Another important component for diagnostics is the Center for Diagnostics. This module loads and maintains the diagnostic network and the states of diagnostics. It also receives information regarding the state of various services. This information is used to update the diagnostic network and perform inferences and trouble shooting. A fully centralized approach to diagnostics, however, would increase the complexity of the system.

So to reduce the complexity of the Diagnostics center we came up with a distributed model of diagnostics. We pre-process information regarding the state of the services and collect the information in a separate state of each service called the diagnostic state. This info can then be directly plugged into the diagnostic manager via a notification system. Thus diagnostics becomes more distributed, thereby reducing the load on the central diagnostics module.

IV. Implementation

A. Model Construction

The tool used for modeling the network is GeNIe [24], developed at the Decision Systems laboratory in the University of Pittsburg. It provides a windows based interface, that allows creation of Bayesian networks in a click and drop interface. GeNIe is the graphical user interface for SMILE [25], a fully platform independent C++ library implementing decision theoretic models such as the Bayesian networks. It allows editing, creating, saving and loading graphical models, and using them for probabilistic reasoning and decision making under uncertainty. We make use of the SMILE.NET wrapper classes for diagnostic programming purposes.

The first step in creating a Bayesian network for the system is to identify the variables in the network and elicit the structure of the network. An initial set of variables were identified based on the life cycle modeling of the services. Then we identified and verified the causal links in the model. By maintaining this causal perspective, we correctly represented the dependence and independence relations.

The Fig. 5 shows a modular perspective of the system initialization phase. The link between each of these modules and the ErrorMode represent a causal link which affects the ErrorMode of the system which in turn determines the initialization success / failure of the system.

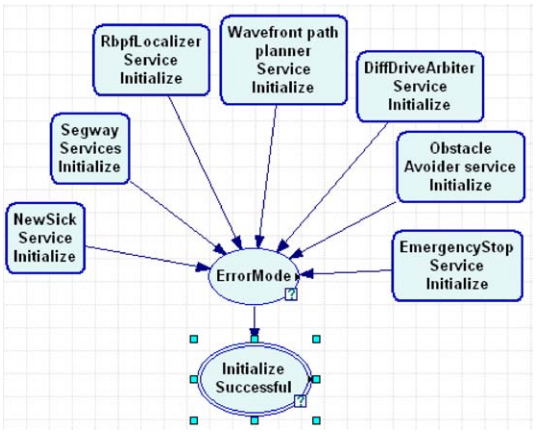

Fig. 5. Status information to be considered for the Initialize Services part of the systems start-up

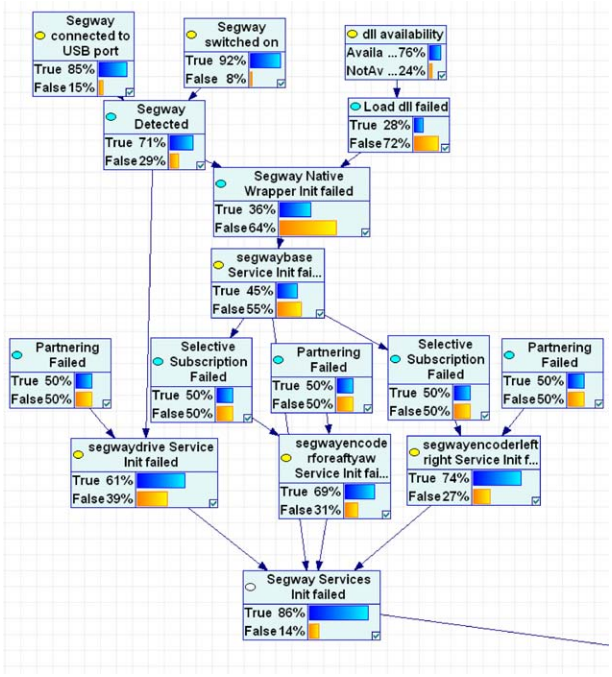

Fig. 6. Segway Services Initialize Network

Now the complexity of designing an initialization phase of the diagnostic model is broken down to the complexity of designing a Initialization diagnostic network for each of the services. The Fig. 6 shows the network modeling of the Segway service initialization network. The Fig. 7 shows the network modeling of the NewSick (Sick laser range finder) service initialization.

Some of these network nodes are set as observation nodes and some are set as target nodes. Observation nodes refer to possible tests, error messages and symptoms. Target nodes refer to possible faults or defective components. One of the states is selected as the target state. Each of the target states of the target nodes are ranked according to the likelihood of their failure. Faults can be tested for in this order. The observation nodes are also ranked and this information is essential for trouble shooting. Trouble shooting is done using this ranking.

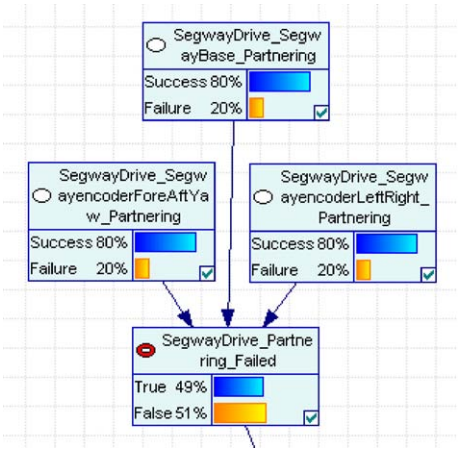

Fig. 8. Partnering Failed expanded for Segway Drive Service

B. Object Oriented Model Development

As can be seen in Fig. 6 and Fig. 7 the network, there are some nodes that are repetitive. These are the abstractions of the sub-network instances and they act like place holders and are the input nodes or output nodes of sub-networks that make up this object oriented Bayesian network. For example consider the place holder node "Partnering failed". During runtime, the parent network is first loaded. Then when inference is performed on the network, the Partnering failed network instance is added to the parent network with the output node set as "Partnering failed". The conditional probability values are also assigned at run time as they are assumed to follow a generic probability distribution. Fig. 8 shows the instantiation of the Partnering failed network for Segway drive service. We make use of SMILE.NET api's to perform this expansion of the network.

C. Quantitative Modeling

The quantitative modeling of a network refers to determining the parameters or the numbers of the networks. This involves estimating the conditional probability values. This step is also called the probability elicitation step. This involves meeting with experts or using statistics gathered over time to figure out the probability values. But for experimental purposes, the probability values have been assigned manually and systematically to avoid any serious diversions from the expected normal behavior of the system. Also for now the cost of troubleshooting has not been included as a diagnosis parameter of the system. In other words all the troubleshooting steps are assumed to have a uniform cost.

D. Diagnostic Manager Service

The center for diagnostics of the navigation system is the diagnostic manager service. The navigation system under consideration has a service oriented architecture. For the diagnostics to be performed in such a system, we create a service called the diagnostic manager service. The purpose of this service is to receive diagnostics and error information from the various services that make up the system via a notification

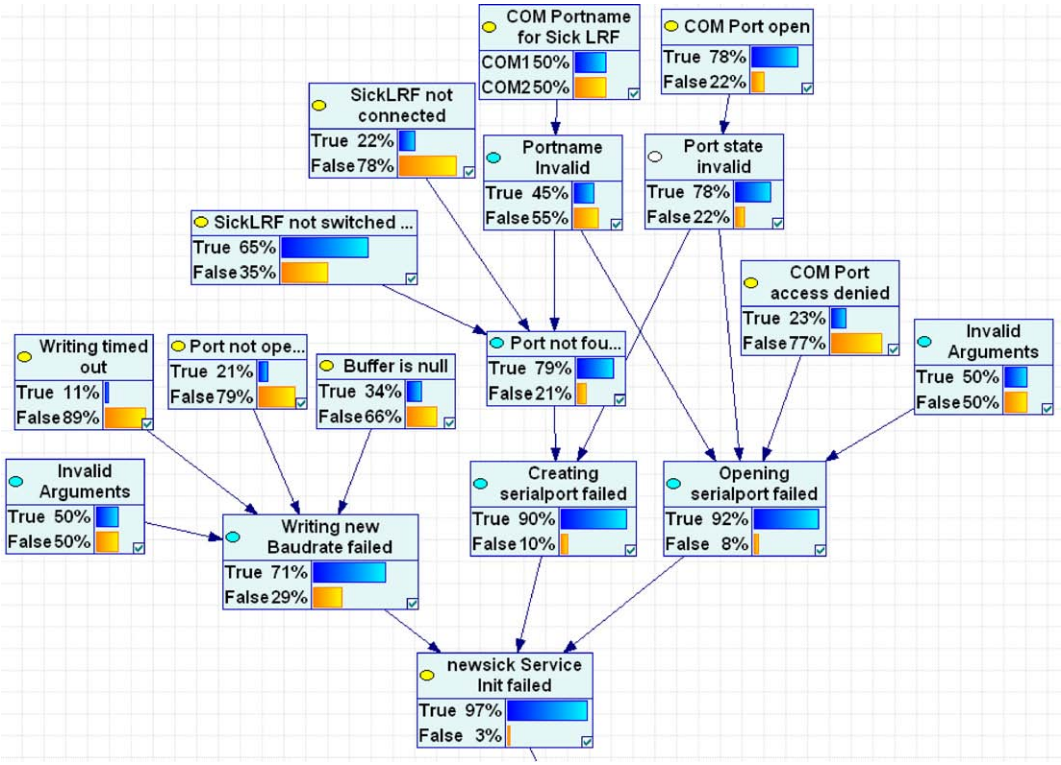

Fig. 7. NewSick Service Initialize Network

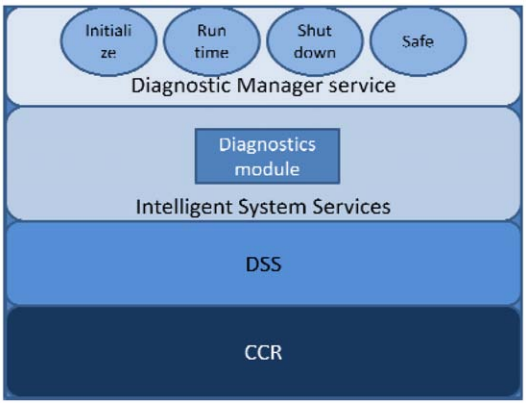

Fig. 9. System Architecture with Diagnostics

system. The diagnostic state/module of each service maintains specific pre-processed information pertaining to error and status tracking for these services.

The diagnostic manager service selectively subscribes to the diagnostic state of the various services. This information is used to update the diagnostic network and state of the service and inferences are performed with this network. If an error symptom is detected the system goes to trouble shooting mode and the cause of these errors are diagnosed through systematic troubleshooting steps identified using the network. The Fig. 9 shows the high level architecture of the system with the integrated diagnostic manager service.

During the system initialization phase, the diagnostic manager service cannot get status information from the services that are being initialized. So access system information on whether a service has been initialized successfully or not, the diagnostic manager service subscribes to the console output service that shows status information on service initialization. Once the component services of the system have been initialized successfully, the diagnostic manager service selectively subscribes to the diagnostic state/module of these services. Notifications are sent to the diagnostic manager when the state of services changes.

While initializing the diagnostic manager service, the diagnostic networks of the various phases of the services of the system are loaded using the SMILE.NET API's. For diagnostic purposes, all the place holder input/output nodes are extended by instances of the sub networks and the conditional probability values are assigned to these nodes. All this is done using the SMILE.NET interface library.

E. Performing diagnostics

During the initialization phase of the system, if there are errors identified in the system, they are entered as evidence into the network. After entering evidence, the probability values are propagated and the network is updated. The subset of the target nodes that may have caused these observable errors are chosen as the possible faults in the system. This subset is retrieved from the causal tree of the network by identifying the dependent nodes.

These target faults are then pursued by the system and relevant tests are requested one by one in the ranked order of importance and the network is updated each time to reflect the result of the tests. In this way, the fault in the system which caused an initialization error is determined and troubleshooting

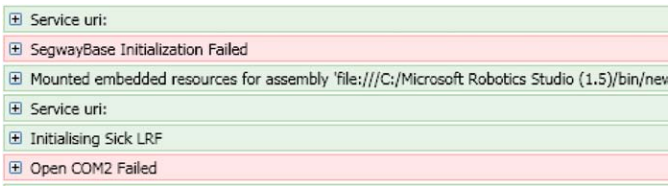

Fig. 10. Console Output Service

is performed. For further details on the SMILE diagnostic network troubleshooting, please see the documentation on SMILE [25].

V. Example Performances

To demonstrate the trouble shooting of the possible navigation system errors, we create a manifest that consists of only those services belonging to the navigation system. We then load the manifest in the dss host to start the navigation system of the mobile manipulator. The diagnostic service is the first module to be initialized. This module initially subscribes to the console output service to detect service initialization errors and then it selectively subscribes to those services that were successfully initialized.

The navigation system was loaded in a PC not connected to the segway or the laser range finder. This results in two errors and they are discussed in section V-A and section V-B. These are initialization errors that can be seen as messages from the console output service as seen in Fig. 10.

A. Error: Segway Base Initialization Failed -2345

When the system detects a "segway base initialization error", it enters the evidence in the network and updates the probability distribution in the causal tree. This error could be caused by three possible faults as observed in the Fig. 6. Once the system is updated with the new error observation, we see that only two of these faults, "Segway native wrapper init failed" and "Load dll failed", have a probability higher than 0.5. The fault that caused this error can be identified by setting these two possible faults as pursued faults and troubleshooting the system. The required evidences are ranked and have to be set by either the diagnostic user or the system. By performing this troubleshooting, the correct fault can be identified and resolved.

B. Error:Opening COM2 Failed

The second error is "COM2 failed error" and this error is identified as the newsick service initialization error as this error is caused by the newsick service initialization state. Fig. 12 shows the possible faults for this error once the system is updated with the new error evidence. The set of trouble shooting steps that the user has to perform are ordered by their ranking. As can be seen the "Open Serial Port failed" fault has the highest probability. During each troubleshooting step, the user's evidence is set and the diagnostic network is updated to reflect this.

Fig. 11. Segway base Service Init Failed Diagnostics

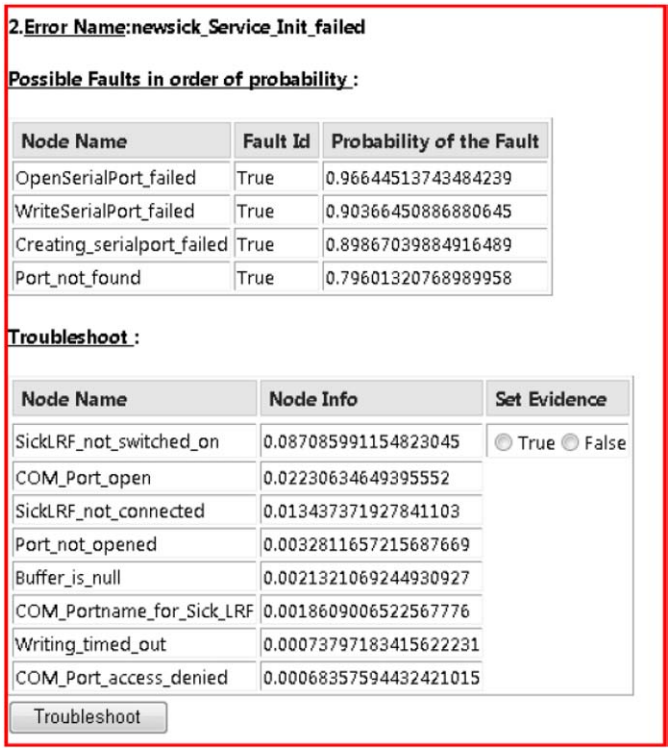

Fig. 12. NewSick Service Init Failed Diagnostics

VI. Summary

Diagnostics is an important aspect of operation for industrial robots to maintain operation over time. Controls built into the robots would enable diagnostic capability in robots. In this paper we presented a plug and play mode of diagnostics in which we create a service (the diagnostic manager) that acts as the center for diagnostics and diagnostic modules that could be plugged into the various robot services to collect information regarding the status of the robot, process these and send back the data in the required format to the diagnostic manager. Diagnostics is therefore distributed across the services. A way to model the errors in the robot system is discussed and some sample implementations are presented. We have discussed an

organized and systematic way of error modeling of complex robot systems.

We plan to enhance this work by implementing error modeling for the other phases of the life cycle and thus develop a complete modeling of the system. There are obvious opportunities for improvement in the area of quantitative modeling where we could assign more reliable probabilistic distributions to the variables by running several experiments on the robot and talking to experts working with the robot everyday. A more elaborate model for probability elicitation could thus be developed. This work would then result in a comprehensive model for intelligent robot diagnostics.

REFERENCES

[1] D. Heckerman, A. Mamdani, and M. Wellman, "Real-world applications of Bayesian networks," *Communications of the ACM*, vol. 38, no. 3, pp. 24–26, 1995.

[2] P. Haddawy, "An Overview of Some Recent Developments in Bayesian Problem-Solving Techniques," *AI Magazine*, vol. 20, no. 2, pp. 11–19, 1999.

[3] C. Skaanning, F. Jensen, U. Kjaerulff, P. Pelletier, L. Jensen, M. Parker, and J. Bogorad, "Automated diagnosis of printer systems using bayesian networks," Jan. 11 2001, uS Patent App. 09/758,891.

[4] L. Console and O. Dressler, "Model-based diagnosis in the real world: lessons learned and challenges remaining," in *International Joint Conference On Artificial Intelligence*, vol. 16. Lawrence Erlbaum Associates Ltd., 1999, pp. 1393–1400.

[5] H. Lin, Y. Yih, and G. Salvendy, "Neural-network based fault diagnosis of hydraulic forging presses in China," *International Journal of Production Research*, vol. 33, no. 7, pp. 1939–1951, 1995.

[6] J. Breese, E. Horvitz, M. Peot, R. Gay, and G. Quentin, "Automated decision-analytic diagnosis of thermal performance in gas turbines," *Proc. Turbine Expo*, vol. 92, 1992.

[7] W. Fenton, T. McGinnity, and L. Maguire, "Fault diagnosis of electronic systems using intelligent techniques: a review," *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, vol. 31, no. 3, pp. 269–281, 2001.

[8] U. Kjaerulff and A. Madsen, *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer Verlag, 2007.

[9] M. Kramer and B. Palowitch, "A rule-based approach to fault diagnosis using the signed directed graph," *AIChE Journal*, vol. 33, no. 7, pp. 1067–1078, 1987.

[10] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *Proc. MLnet Summer School on Machine Learning and Knowledge Acquisition*, pp. 1–58, 1994.

[11] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *International Conference on Autonomic Computing*, 2004.

[12] C. Skaanning, F. Jensen, and U. Kjaerulff, "Printer Troubleshooting Using Bayesian Networks," *Lecture notes in Computer Sciences*, pp. 367–379, 2000.

[13] K. Przytula and D. Thompson, "Construction of Bayesian networks for diagnostics," in *Aerospace Conference Proceedings, 2000 IEEE*, vol. 5, 2000.

[14] K. Przytula, F. Hagen, and K. Yung, "Bayesian networks for satellite payload testing," in *Proceedings of SPIE*, vol. 3812. SPIE, 1999, p. 225.

[15] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 256–265.

[16] A. Madsen, M. Lang, U. Kjaerulff, and F. Jensen, "The Hugin Tool for Learning Bayesian Networks," *Lecture notes in Computer Sciences*, pp. 594–606, 2003.

[17] S. Andersen, K. Olesen, F.V.Jensen, and F. Jensen, "Hugin, A Shell for Building Bayesian Belief Universes for Expert Systems," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, vol. 2, 1989.

[18] E. Horvitz, D. Hovel, and C. Kadie, "MSBNx: A Component-Centric Toolkit for Modeling and Inference with Bayesian Networks," Technical Report MSR-TR-2001-67, Microsoft Research, Redmond, WA, July 2001, Tech. Rep.

[19] M. Druzdzel, "GeNIe: A development environment for graphical decision-analytic models," in *Proceedings of the 1999 Annual Symposium of the American Medical Informatics Association (AMIA-1999)*, 1999.

[20] ——, "SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models," in *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference*. American Association for Artificial Intelligence Menlo Park, CA, USA, 1999, pp. 902–903.

[21] *Microsoft Robotics Studio 1.5*, Microsoft Corporation, 2009. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb483104.aspx

[22] H. Christensen and P. Case, "Mobile manipulation for everyday environments," in *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, 2008.

[23] F. Jensen, *Bayesian Networks and Decision Graphs*. Springer Verlag, New York, NY, 2001.

[24] *GeNIe Documentation*, S. o. I. S. U. o. P. Decision Systems Laboratory (DSL), 2008. [Online]. Available: {http://genie.sis.pitt.edu/wiki/GeNIe_Documentation}

[25] *SMILE Documentation*, S. o. I. S. U. o. P. Decision Systems Laboratory (DSL), 2008. [Online]. Available: {http://genie.sis.pitt.edu/wiki/SMILE_Documentation}