A Systems Approach towards Developing a Diagnostic System for Complex Robots

Rahul Krishnan
Worcester Polytechnic Institute
Contact Information
rkrishnan2@wpi.edu

Shamsnaz Virani Bhada
Worcester Polytechnic Institute
Contact Information
ssvirani@wpi.edu

Abstract. Present day robots are highly complex systems. Most robots perform tasks that are time-critical in environments where human intervention may not be possible. When a failure occurs in such situations, diagnosing it can be a major challenge. To tackle this, a methodology to develop a diagnostic system using a systems approach is presented in this paper. The use of SysML diagrams as a tool to understand system behavior is highlighted. Finally, this methodology is used to build the diagnostic system of a complex humanoid robot.

Introduction

In a number of applications of robot systems, diagnosing failures through human intervention is either slow, unreliable, expensive or impossible (Verma, Gordon and Thrun, 2004). Such applications could be space exploration, mining, search & rescue and nuclear waste cleanup. These robots are comprised of several sub-systems – robot platform (or chassis), manipulation, control system, power, wireless communication and sensing, that are in-turn made up of numerous hardware and software modules (Carlson and Murphy, 2003) (Murphy et. al, 2009). For example, the PR2, a mobile manipulation robot, has approximately 30 sensors that each interact with the control system, the power system, the network system and with each other (Willow Garage, 2012). The increase in system complexity comes from the number of sub-systems and their inter-dependence. As complexity grows, so does the difficulty in detecting and identifying failures in the system, primarily due to the large space of possible failures (Verma, Gordon and Thrun, 2004). In a study that spanned over three years, it was shown that the overall Mean Time Between Failures (MTBF) in mobile robot systems is around 24 hours (Carlson, Murphy and Nelson, 2004). The common source of failures were the control system and the mechanical platform. Similar results were found in a previous study by Carlson and Murphy (2003). Carlson et al. (2004) also found that the average availability of the robots studied was 54% and average downtime was 23.2 hrs. Almost an entire day of robot operation is lost trying to identify and fix the failure, resulting in a loss of productivity. Allowing robots to autonomously detect and identify faults can help reduce the average downtime. This functionality can be implemented in robots by adding a diagnostics sub-system to the system architecture. It will be responsible for continuously monitoring the system and diagnosing failures when they occur.

A key component of diagnostic systems is the diagnostic strategy it implements. A diagnostic strategy is the method employed to detect faults or failures and subsequently diagnosing their cause (Katipamula and Brambley, 2005). They are broadly classified based on the type of information it uses to detect and diagnose faults or failures. Faults refer to an unpermitted deviation of at least one characteristic property of the system from an acceptable behavior (Isermann, 2005). Whereas, failures refer to a system condition where a complete loss of service availability occurs. They are often employed interchangeably in literature. A review of fault detection and diagnostic strategies by Venkatasubramanian et al. (2003) and Katipamula and Brambley (2005) tells us that diagnostics are
either based on prior knowledge of the system, obtained through experience and subject matter experts, or are completely data driven. Approaches that use prior knowledge are classified as model based approaches, where the model is used to calculate expected values of the system. These values are compared to the actual measurement data to evaluate the condition of the system. Data driven approaches, which are classified as process history based approaches, extract the behavior of the system from measurement to build a model of the system.

![Figure 1. Classification of Fault Detection and Diagnosis Strategies](image)

Figure 1 shows the classification scheme for fault detection and diagnostic strategies (Katipamula and Brambley, 2005). In this paper, a model based approach is explored by developing a rule based expert system. Rule-based diagnostic systems detect and identify faults in accordance with the rules representing the relation of each possible fault with the actual monitored condition (Yam et al., 2001). The strengths of rule based expert systems lies in its ease of development, straightforward reasoning, the ability to handle uncertainty, and the ability to provide clear explanations for conclusions reached (Venkatasubramanian et al., 2003). An example of a rule based diagnostic system can be seen in Luskay et al. (2003), where rules are implemented in a decision tree structure. Alternatively, bayesian network models have emerged as a favored technique for diagnostic applications. Chandrababu and Christensen (2009) provided a systematic methodology for modeling errors in a robot using a bayesian model. Onisko et al. (2001) compared ruled based and bayesian network based approaches in medical diagnostic systems and found that both have their own merits. Their conclusion was that rule based systems allows for testing models that follow the trace of system’s reasoning, whereas probabilistic systems (like bayesian networks) allows for models to be trained and fine-tuned based on existing data sets. For this paper, we chose to explore rule based diagnostic systems. Future iterations of this project will involve developing a diagnostic system using bayesian networks.

While considerable research has been done in the area of diagnostic strategies, no literature is available on the process of incorporating them into the diagnostic system of a robot. In this paper, we propose a methodology that utilizes a systems approach towards developing diagnostic systems that could help reduce the average downtime of a robot and reduce the need for human intervention in diagnosing robot failures. To achieve this, we translate expert knowledge of system behavior into the building blocks of the diagnostic system. This translation is done using SysML diagrams, which capture the structural and behavioral aspects during the initialization and runtime phases of the robot. A detailed description of the robot used for this paper is provided in the next section.

**System Definition**

In this section we discuss the complex system for which the diagnostic system is designed. The first section provides a brief description of the system and its underlying architecture. The second section gives a quick overview of the workings of the software framework of the robot.
The ATLAS Robot. The system under consideration for diagnostics is Boston Dynamics’ Atlas Robot, a humanoid robot that has 30 degrees of freedom, weighs approximately 180 Kg, and is 1.8m tall (DeDonato et. al, 2017). Its lower limbs, torso and upper arms are hydraulic joints while the forearms and wrists are electronically powered (DeDonato et. al, 2017). The robot has a Carnegie Robotics’ MultiSense sensor, featuring a stereo camera and a rotating LIDAR (Light Detection and Ranging) for perception, as well as additional head cameras. The MultiSense sensor is used to provide high-resolution, high-data-rate, and high-accuracy 3D range data in the form of 3D point clouds. The robot has three on-board Intel i-7 processors. These computers run the distributed Robot Operating System (ROS). ATLAS robot uses ROS to implement the entirety of the software it governs.

Figure 3 shows the system architecture of the robot. The primary external sensors on the robot are the E-BOX, which is a physical emergency stop button on the robot, the MultiSense sensor, SA-Cameras, (additional head cameras) and electronically controlled hands. Each of them are connected to one of the on-board computers. Each computer has a dedicated task assigned to it. CPU_A deals with balancing and motion control algorithms. CPU_B is interfaced with the MultiSense sensor and the SA cameras. This computer runs all of the drivers and algorithms for the cameras and image processing. Finally CPU_C is connected to the hands, and is responsible for operations involving the hands. Additionally, it also handles communication to the field computer over a wireless network. To do this, the ROS messages are first converted into User Datagram Protocol (UDP) packets. The field computer has the task of converting the UDP packets from the on-board computer back into ROS.
messages. These ROS messages are then sent to the Graphical User Interfaces (GUIs) for the operator to view or to other nodes for further processing.

![Node Publication Topic Subscription](image)

Figure 4. Core components of ROS

**ROS definition.** Another important component of the robot is the Robot Operating System (ROS). It is an open source, middleware that provides a structured communications layer above the host operating systems of a heterogeneous compute cluster (Quigley et. al, 2009). It provides services like hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. Additionally, it also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. At the lowest level, ROS offers a message passing interface that provides inter-process communication. In ROS terminology, the processes are called Nodes, which perform computations of some form. A robot control system usually comprises of several nodes. The nodes could represent hardware components like sensors and actuators. It can also represent software components that perform path planning, robot localization, image processing or provide graphical views of the system. Nodes communicate with each other by passing ‘messages’, which is a data structure comprising of typed fields (integer, floating point, boolean, etc.). ROS provides a transport system to exchange information between nodes using publish/subscribe semantics. Nodes transmit a message by publishing it to ‘topics’, which are named buses used to identify the content of messages. Each topic or bus has a name and any other node can connect to it to transmit or receive messages. Nodes that want to receive a certain type of information subscribe to the appropriate topic. Figure 4 illustrates the core components of ROS.

**Methodology**

- **Modeling**
  - System structure - Block Definition Diagram
  - Sub-system behavior - Interface diagram

- **Defining**
  - Verification tests
  - Sub-system diagnostic state

- **Selecting**
  - Monitoring method
  - Diagnostic strategy

Figure 5. Methodology: Systems approach to developing diagnostic systems

In this section, we describe the proposed methodology to develop a diagnostic system for complex robots. The first step involves modeling the system and sub-systems using SysML diagrams. The second step deals with defining both, tests to check nominal operation and diagnostic states of sub-systems. In the last step, a monitoring method and diagnostic strategy is chosen.


Modeling System Structure

The first step is to understand the robot system architecture. This helps us identify all the sub-systems of the robot and how they communicate with each other. The key outcome of this step is to capture the structural and behavioral aspects of the system. Block Definition Diagrams (BDD) and interface diagrams are excellent tools for such a task. A BDD provides a black box representation of the robot system, with the hierarchy of its composite blocks. This includes both hardware and software components of the system. It also shows the different relationships that exist between system blocks. An interface diagram shows the interaction points between blocks or 'components’. These interaction points help us visualize the information being exchanged between blocks as well as its direction. This information is very useful for identifying the dependencies between sub-systems. The BDD of the ATLAS robot can be seen in figure 6.

![Block Definition Diagram of ATLAS robot](image)

Figure 6. Block Definition Diagram of ATLAS robot

Modeling Sub-systems

Once the system is modelled, the next step is to identify and further analyze the sub-systems. They are typically sensors or actuators, i.e., a sub-system that has some output associated with it. For example, in the ATLAS robot, the MultiSense sensor is a sub-system that provides 3D range data and live images. On the other hand, the on-board computers (CPU_A, CPU_B and CPU_C) do not fall under the same category. By identifying the output for each sub-system you can elicit its dependency with other sub-systems. For the condition where the output is not available, the interface diagram can be used to provide a comprehensive list of points where something could have ‘gone wrong’. Figure 7 shows the interface diagram of the ATLAS robot. The interface diagram helps us understand the behavioral aspects of the sub-system. It shows how the behavior of a sub-system is influenced by its interaction with other sub-systems. These behaviors are identified based on the operational state of
the robot – initialization or runtime. In the initialization phase, different sub-systems of the robot are setup or configured to the appropriate network. This includes loading sensor drivers and configuring computers over the network for bi-directional communication. The second phase is runtime during which specific functions of the sub-system are executed. In the context of robots and ROS, the sensors will transmit information to ROS nodes, which will either perform some computation, publish the data or subscribe to another node. By identifying sub-system behaviors as initialization or runtime, the type of interaction and information flow is better understood.

![Figure 7. Interface Diagram of ATLAS robot](image)

**Tests to Verify Normal System Operation**

The interface diagram created in the previous step can be used as a basis for writing test cases to verify that the links between sub-systems are operating correctly. These test cases can be used to check both software and hardware modules. If the test reports an error in the module, it returns a flag for the diagnostic system and/or a readable summary of the error for the user.

**Diagnostic State to Monitor Sub-System**

A diagnostic state is defined for each sub-system. It is used as a flag to inform the diagnostic system if a sub-system is operating as expected or has encountered a failure. What constitutes as ‘normal’ operation must be defined for each sub-system, by defining boundary conditions. When the data returned by a sub-system falls outside this boundary, it triggers an ‘ERROR’ state. The diagnostic states are defined as: NORMAL and ERROR. When an ERROR is encountered, the diagnostic system is alerted to identify the cause of the failure.

**Diagnostic Strategy**

In our approach, a diagnostic software package is added to the software architecture of the robot. It is responsible for monitoring the diagnostic states of the other sub-systems and troubleshooting when ERROR state is found. The key decisions that need to be made when designing the diagnostic system are:

- Monitoring method – The diagnostic system receives information about the state of other sub-systems. An important design decision is the frequency at which it receives this information. All sub-systems will publish its diagnostic state information at a certain fixed sampling rate. The diagnostics system will subscribe for this information, and verify ‘NORMAL’ operation.
If an ‘ERROR’ state is found, the robot is paused and the diagnostic system begins to troubleshoot the error.

- Diagnostic strategy for troubleshooting – Based on the classification scheme in Fig 1, a diagnostic strategy must be chosen. It will use the test cases that were obtained from the interface diagrams to elicit the root cause of the failure. For the ATLAS robot, we have used the rule based system. This is described in detail in the next section.

**Case Study: ATLAS Robot**

In this section, we demonstrate the application of the proposed methodology to build a diagnostic system for the ATLAS robot using the rule based diagnostic strategy.

The first step is to build a BDD and interface diagram of the system, as illustrated in figure 6 and figure 7. We selected a subset of this system and built models of the diagnostic strategy for the MultiSense sensor sub-system. As described earlier, the MultiSense sensor provides RGB images as well as 3D point clouds from a laser scanner. A failure of the MultiSense sensor sub-system would be when no image data is received at the field computer. Essentially, the data received is null. This could be because of problems with the physical connection, sensor configuration or initialization errors in ROS. From figure 7, the relationship between the MultiSense sensor and the other sub-systems is understood. The sub-systems that could potentially cause a failure include – the MultiSense driver, CPU_B and ROS running on CPU_B. Since all CPUs are connected to the same network, a problem in CPU_A, CPU_C and the communication with field computer can also cause a failure. The next step is to identify test cases that can be used to verify that the interactions between the identified sub-systems are working correctly. Each test case is written based on whether the interaction takes place during the initialization or runtime operational state of the robot. When the diagnostic system receives invalid image data from the MultiSense sensor, the diagnostic state switches from NORMAL to ERROR. The last step is to select a diagnostic strategy of the diagnostic system. Below is the implementation of the rule based strategy for the MultiSense sensor of the ATLAS robot.

**Rule Based Strategy.** Figure 8 illustrates the decision tree for the rule based strategy during the initialization operating state of the robot. These rules have been developed by prioritizing the test cases developed earlier. Priority is set based on the opinion of subject matter experts who have significant experience with the robot. The decision tree traces how they would go about identifying faults manually. When the MultiSense sensor sub-system reports an ERROR diagnostic state, the diagnostic system beings to troubleshoot the failure based on the decision tree shown in Figure 8. Each test returns a binary pass or fail result. The first test determines if the failure was caused due to the sensor or because of the communication channel between the sensor and field computer. Based on the result, the appropriate branch of the tree is traversed until the failure is identified.

**Summary and Future Applications**

In this paper we presented a methodology to develop a diagnostic system for complex robots. The diagnostic system of the robot routinely monitors various sub-systems and identifies the failure when it occurs. This system is built using a suitable diagnostic strategy. The selected strategy dictates the process by which the failure is identified. The test cases built for each sub-system helps narrow down the root cause of the failure. A model of the rule based diagnostic strategy was built for the diagnostic system of the ATLAS robot. The next phase of this research is to incorporate the model of the diagnostic system into the ATLAS robot and test its accuracy. There’s also an opportunity to develop a more complete model of the system, by adding other sub-systems into the diagnostic strategy as well. This would result in a more comprehensive diagnostic system of the ATLAS robot. Additionally, diagnostic strategies using probabilistic systems (like bayesian networks) can also be explored.
Figure 8. Overview of decision tree for rule based diagnostic strategy

References


Biography

Rahul Krishnan

Rahul Krishnan is a PhD student at Worcester Polytechnic Institute. His focus is on diagnostics strategies, robotics, and Model Based Systems Engineering (MBSE). Rahul has completed his Master’s degree in Robotics Engineering at Worcester Polytechnic Institute too.

Shamsnaz Virani Bhada

Shamsnaz Virani Bhada, Assistant Teaching Professor of Systems Engineering at Worcester Polytechnic Institute, earned her Ph.D.in Industrial and Systems Engineering from The University of Alabama at Huntsville. Dr. Virani’s research interests include Modeling Based Systems Engineering (MBSE), Engineering Education and Team Mental Models. She is founding member of Empowering Women as Leaders in Systems Engineering (EWLSE), International Council of Systems Engineers (INCOSE), American Society of Engineering Education (ASEE) and Institute of Industrial and Systems Engineers (IISE).