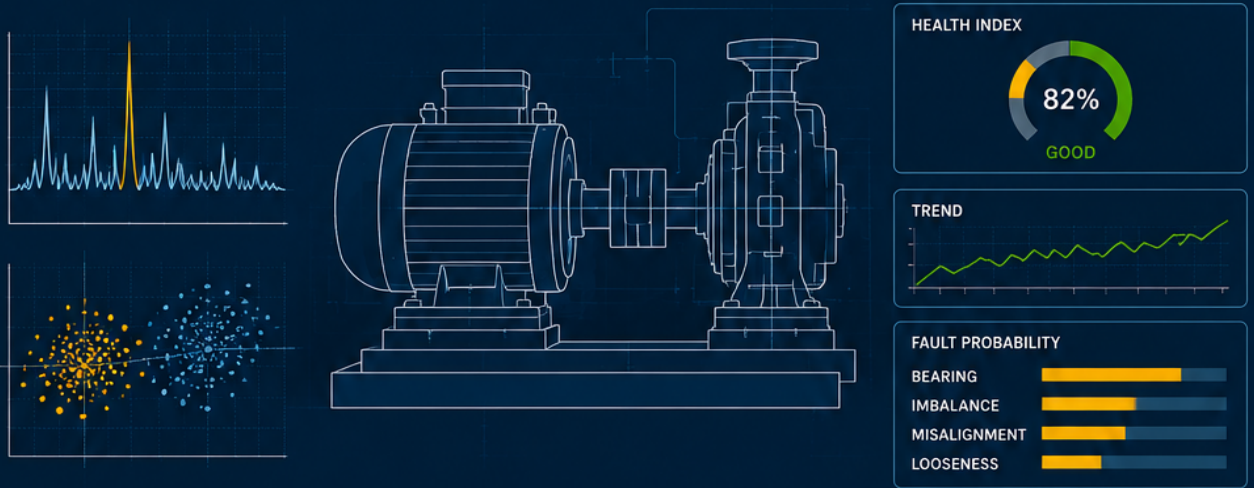
DIAGNOSTICS ENGINEERING

PRINCIPLES, METHODS, AND APPLICATIONS

A Comprehensive Guide to
Fault Detection, Condition Monitoring,
and Health Management Systems

SIGNAL PROCESSING | SYSTEM MODELING | DATA SCIENCE | FAULT DETECTION | HEALTH MANAGEMENT



HARVEY C.

Diagnostics Engineering: Principles, Methods, and Applications

A Comprehensive Guide to Fault Detection, Condition Monitoring, and Health Management Systems

Steve T. Publications

This book is available at https://leanpub.com/diagnosticsengineering

This version was published on 2026-07-13

Contents

A Comprehensive Guide to Fault Detection, Condition Monitoring, and Health Management Systems

Diagnostics engineering sits at the intersection of signal processing, system modeling, data science, and reliability theory. This book takes you from first principles through advanced techniques, covering the full lifecycle of diagnostic systems for complex engineering applications. Whether you are designing condition monitoring for rotating machinery, implementing fault detection in automotive ECUs, building predictive maintenance platforms for industrial assets, or validating safety-critical diagnostic software, this book provides the theoretical foundations, practical engineering guidance, and real-world case studies you need. Every method is explained not just in terms of how it works, but why it works, what trade-offs are involved, and how to avoid common pitfalls. The goal is to equip you with both deep understanding and deployable expertise.

Introduction

On July 6, 1996, an Ariane 5 rocket lifted off from Kourou, French Guiana, and for thirty-two seconds it climbed exactly as designed. Then something went wrong. The inertial reference system overflowed a 16-bit integer during a course correction computation, generating a data exception that the software did not handle. The flight computer discarded all navigation data, causing the attitude platform to misalign by more than forty-five degrees. The rocket began tumbling. Twenty seconds after liftoff, the self-destruct system detonated, and the $500 million vehicle disintegrated over the Atlantic Ocean.

The root cause was not a hardware failure. It was a software exception that should have been caught by diagnostics but was not. The Ariane 5 inherited its flight software from Ariane 4, where the same computation never overflowed because the flight profile was different. No diagnostic mechanism existed to detect that the navigation system had lost valid data before the self-destruct sequence triggered. This failure became a textbook case study in the consequences of inadequate fault detection and diagnostic coverage [1].

Consider another scenario, less dramatic but far more common: a centrifugal pump in a chemical plant runs for eighteen months before its bearing begins to show subtle vibration changes at 4,200 Hz. A condition monitoring system with properly tuned envelope analysis picks up the high-frequency impact pattern characteristic of an outer race defect three months before catastrophic failure would have occurred. The maintenance team schedules a replacement during the next planned shutdown, avoiding $250,000 in emergency repair costs and four days of unplanned production loss.

These two stories illustrate the full spectrum of diagnostics engineering. At one extreme, we have safety-critical systems where failure means catastrophe, demanding rigorous fault detection, isolation, and recovery (FDIR) architectures validated against strict certification standards. At the other extreme, we have industrial assets where early fault detection translates directly to cost savings through predictive maintenance. Both require systematic diagnostic capability, but the design trade-offs, validation requirements, and operational constraints differ enormously.

Diagnostics engineering is the discipline that makes both of these outcomes possible. It encompasses everything from selecting the right accelerometer for vibration monitoring, through designing residual generators that distinguish real faults from sensor noise, to implementing machine learning classifiers that generalize across operating conditions and deploying diagnostic protocols that communicate fault information reliably between distributed systems.

What This Book Covers

This book provides a comprehensive treatment of diagnostics engineering organized around the full lifecycle of a diagnostic system. We begin with foundational concepts: what diagnostics means in an engineering context, how faults are classified, and the reliability theory that underpins all diagnostic design decisions. From there we move through the technical core of the discipline.

System modeling forms the bedrock of model-based diagnostics. You will learn to build mathematical models of physical systems using first-principles approaches like state-space representations and bond graphs, as well as data-driven system identification methods. These models serve as the reference against which actual system behavior is compared to detect anomalies.

Sensor technologies and signal acquisition address the hardware foundation: how physical phenomena are sensed, conditioned, digitized, and made available for analysis. This includes transduction mechanisms, signal conditioning circuitry, sampling theory, and data acquisition system architecture.

Signal processing provides the mathematical toolkit for extracting diagnostic information from raw sensor data. Time-domain statistics, Fourier spectral analysis, wavelet transforms, envelope detection, and order tracking methods are covered with engineering intuition about when and why each technique applies.

Model-based fault detection covers analytical redundancy approaches that use system models to generate residuals – the core mechanism by which model-based diagnostics detect faults. The parity space approach, observer-based methods using Luenberger observers and Kalman filters, parameter estimation techniques, and the fundamental robustness versus sensitivity trade-off are all treated in depth.

Data-driven and machine learning diagnostics address modern approaches that learn diagnostic patterns directly from data. Supervised classification, unsupervised anomaly detection, deep learning architectures including CNNs and RNNs, and the practical challenges of training data quality, class imbalance, and domain adaptation are covered systematically.

Condition monitoring and health management systems show how individual diagnostic algorithms integrate into operational systems for continuous monitoring, threshold design, alarm management, health index construction, and remaining useful life estimation.

Embedded diagnostics address the constraints of implementing diagnostic capability in resource-limited real-time systems: built-in self-test architectures, watchdog timers, memory error correction, and hardware-level diagnostic mechanisms.

Automotive diagnostics provides a deep dive into one of the most mature application domains, covering OBD-II standards, Diagnostic Trouble Code management, Unified Diagnostic Services protocol, CAN bus diagnostics, and powertrain monitoring systems.

Industrial diagnostics and predictive maintenance cover applications in manufacturing, process industries, and infrastructure, including condition monitoring standards, rotating machinery diagnostics, valve stiction detection, and the economic analysis that justifies diagnostic system investment.

Software diagnostics and fault-tolerant computing address error detection and recovery in software systems, including assertions, contracts, redundancy patterns, checkpointing mechanisms, communication protocol integrity checks, and formal verification methods.

Validation, verification, and testing methodologies cover how to prove that a diagnostic system actually works: performance metrics, fault injection testing, test data generation, ROC curve analysis, and the certification requirements of standards like DO-178C, ISO 26262, and IEC 61508.

Finally, emerging trends explore technologies reshaping the field: AI-assisted diagnostics moving toward autonomous diagnosis, digital twins at scale for fleet-wide health management, edge computing architectures, federated learning, and next-generation sensor technologies.

How to Use This Book

This book is designed for readers ranging from advanced undergraduate engineering students to practicing engineers who need deep understanding of diagnostic systems. Each chapter builds on previous material, so reading in sequence is recommended. However, experienced practitioners may want to jump directly to chapters relevant to their application domain.

Worked examples throughout the book demonstrate concepts from first principles through complete implementation. These are not abbreviated sketches but fully explained scenarios with all intermediate steps shown. Comparison tables summarize trade-offs between competing approaches. Checklists provide quick reference for design decisions. Formulas include derivations where they illuminate understanding rather than simply stating results.

Cross-references connect related concepts across chapters. When a concept introduced in an early chapter reappears in a later context, the earlier treatment is referenced so you can trace the development of ideas throughout the book.

A Note on Terminology

Diagnostics engineering draws terminology from multiple disciplines – control theory, signal processing, reliability engineering, data science, and systems engineering. This book uses consistent terminology throughout and defines terms at first use. Where different communities use different words for the same concept (for example, "fault" versus "defect" versus "anomaly"), the preferred term is stated and alternatives noted.

The distinction between fault, error, and failure deserves special attention because confusion among these terms leads to design mistakes. A fault is a physical condition or deviation from specification – a cracked bearing race, a shorted capacitor, a software bug. An error is the manifestation of a fault in system behavior – elevated vibration amplitude, incorrect sensor reading, wrong control output. A failure occurs when the system can no longer perform its required function – seized bearing, lost power,□□ control surface. Diagnostics engineering primarily targets fault detection and isolation; the prevention of errors and failures follows as a consequence.

By the end of this book, you will understand not only how to implement diagnostic systems but how to think about diagnostics at the architectural level: what diagnostic capability is appropriate for a given application, what trade-offs are unavoidable, how to validate that your system works, and how emerging technologies will reshape the field in the coming decade.

Chapter 1: Foundations of Diagnostics Engineering

What Is Diagnostics Engineering?

Diagnostics engineering is the systematic discipline of designing, implementing, and validating systems that detect, locate, characterize, and respond to faults in engineered systems. It spans multiple technical domains – signal processing for extracting diagnostic information from sensor data, mathematical modeling for generating analytical redundancy, machine learning for pattern recognition in complex data, reliability theory for quantifying failure risk, and real-time computing for deploying diagnostic capability in operational environments.

The term "diagnostics" derives from medical diagnosis: the process of identifying a disease by examining symptoms. In engineering, the analogy holds remarkably well. A system exhibits symptoms (elevated vibration, temperature rise, performance degradation), and the diagnostic system must identify which fault is responsible, estimate its severity, and often predict how much time remains before failure. The key difference is that engineered systems are designed with diagnostics in mind from the outset – sensors are placed strategically, redundant measurements are built in, and computational resources are allocated for analysis.

Diagnostics engineering differs from related disciplines in important ways. Testing verifies that a system meets its specifications under controlled conditions; diagnostics operates continuously during field use to detect when the system deviates from normal behavior. Monitoring observes system state without necessarily making diagnostic decisions; diagnostics goes further by classifying observed deviations into specific fault categories. Maintenance schedules repairs based on time or usage; diagnostics provides the information that makes condition-based and predictive maintenance possible.

The scope of diagnostics engineering encompasses several layers:

- **Component-level diagnostics**: Detecting faults in individual components such as bearings, valves, sensors, and electronic circuits
- **Subsystem-level diagnostics**: Identifying faults within functional groups like powertrain systems, hydraulic circuits, or control loops
- **System-level diagnostics**: Assessing overall system health and performance degradation across interacting subsystems
- **Fleet-level diagnostics**: Aggregating diagnostic information across multiple similar assets to identify common failure modes and optimize maintenance strategies

Each layer has different requirements for data granularity, computational resources, response time, and decision authority. Component-level diagnostics may operate at millisecond timescales with simple threshold comparisons. Fleet-level diagnostics may use hours or days of aggregated data and complex statistical models running in cloud environments.

The Diagnostic Lifecycle: Detection, Isolation, Identification, Prognosis

The diagnostic process follows a structured lifecycle often called FDIR – Fault Detection, Isolation, and Recovery – though modern frameworks extend this to include fault identification (severity estimation) and prognosis (remaining useful life prediction). Understanding each stage and its requirements is fundamental to designing effective diagnostic systems.

Fault detection answers the question: "Has something gone wrong?" This is a binary decision problem. The system is either operating normally or it is not. Detection relies on generating a diagnostic signal – called a residual in model-based approaches or an anomaly score in data-driven methods – and comparing it against a threshold. When the diagnostic signal exceeds the threshold, a fault is declared detected.

The fundamental challenge in detection is balancing two competing requirements: sensitivity and robustness. A highly sensitive detector catches small faults early but generates false alarms from normal operating variations, sensor noise, and unmodeled dynamics. A robust detector ignores benign variations but may miss incipient faults until they grow large enough to trigger the alarm. This trade-off lies at the heart of virtually every diagnostic design decision.

Fault isolation answers: "What specifically has gone wrong?" Once detection confirms that a fault exists, isolation identifies which component or parameter is faulty. For example, if vibration monitoring detects abnormal vibration in a motor-pump assembly, isolation determines whether the fault is in the motor bearings, the pump impeller, the coupling, or the mounting structure.

Isolation typically uses multiple diagnostic signals with different sensitivity patterns. Different faults affect different measurements in characteristic ways. By analyzing which residuals are active and their relative magnitudes, the isolation logic can narrow down the fault location. This requires careful design of the residual generation stage so that each residual is sensitive to specific fault types while remaining robust to others – a concept known as structured residual generation.

Fault identification (sometimes called fault estimation or severity assessment) answers: "How bad is it?" Identification quantifies the magnitude, type, and progression rate of the detected and isolated fault. A bearing defect might be classified as "outer race spalling, early stage" versus "outer race spalling, advanced stage." An electrical insulation fault might have its remaining dielectric strength estimated in megohms.

Identification is critical for maintenance decision-making. A small incipient fault may warrant continued monitoring with the next scheduled maintenance window. A rapidly progressing fault demands immediate intervention. Without identification, every detection triggers the same response, leading either to unnecessary maintenance or missed critical failures.

Prognosis (or remaining useful life estimation) answers: "How much time is left?" Prognostics predicts the future degradation trajectory of a faulty component and estimates when it will reach a failure threshold. This goes beyond identifying current fault severity to forecasting how that severity will evolve under expected operating conditions.

Prognostics is fundamentally more uncertain than diagnostics because it must predict future behavior based on past and present observations. Multiple degradation paths may be consistent with current data, and operating conditions may change in unpredictable ways. Prognostic outputs are therefore expressed probabilistically, typically as a probability distribution over remaining useful life rather than a single point estimate.

Recovery (or fault tolerance) answers: "What do we do about it?" Recovery

encompasses the actions taken after diagnosis to maintain system functionality despite the fault. Options range from switching to redundant equipment, through reconfiguring system parameters to compensate for degraded performance, to entering a safe mode that limits functionality while preventing damage.

The FDIR lifecycle is not always executed sequentially. In some architectures, detection and isolation are combined into a single step. In others, prognosis runs continuously in parallel with detection, maintaining an updated health estimate regardless of whether a fault has been formally declared detected. The architecture depends on the application requirements for response time, computational resources, and decision granularity.

Figure 1-1 illustrates the FDIR lifecycle as a flow diagram:

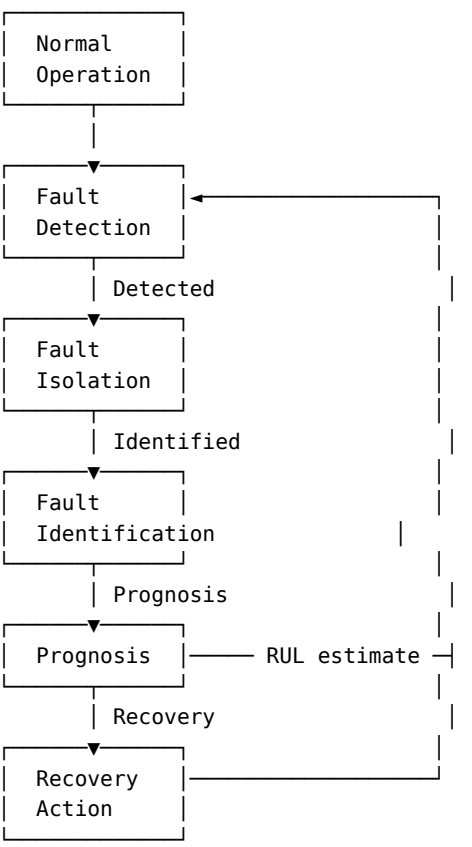


Figure 1-1: The FDIR lifecycle showing the sequential flow from detection through recovery, with feedback loops for continuous monitoring.

Fault Taxonomies and Classification Schemes

Understanding how faults are classified is essential because different fault types demand different diagnostic approaches. A one-size-fits-all diagnostic system does not exist; the choice of sensors, signal processing methods, model complexity, and decision thresholds all depend on the fault characteristics expected in the application.

Faults can be classified along several dimensions:

By temporal behavior:

- **Abrupt faults**: Sudden step-like changes in system behavior, typically caused by instantaneous events like component fracture, electrical short circuits, or complete sensor failure. Abrupt faults are generally the easiest to detect because they produce large, immediate deviations from normal behavior. A broken drive shaft, a blown fuse, or a disconnected sensor cable all manifest as abrupt faults [2].
- **Incipient (gradual) faults**: Slowly developing degradation that manifests as drifting or ramp-like changes in system parameters. Examples include bearing wear, lubricant degradation, valve stiction buildup, and insulation aging. Incipient faults are the most challenging to detect because their early-stage signatures are small and can be confused with normal operating variations [3].
- **Intermittent faults**: Faults that appear and disappear unpredictably, often caused by loose connections, marginal components near failure threshold, or environmental effects like thermal expansion creating intermittent contact. Intermittent faults are notoriously difficult to diagnose because they may not be present during testing but cause failures in operation [4].

By physical nature:

- **Additive faults**: Faults that add an offset or noise component to the system output without changing the underlying dynamics. Sensor bias, actuator stiction, and measurement noise fall into this category. Additive faults are generally easier to handle mathematically because they affect residuals linearly [2].

- **Multiplicative (parametric) faults**: Faults that change system parameters, thereby altering the dynamic behavior. Loss of stiffness in a mechanical structure, increase in electrical resistance due to corrosion, or degradation of heat transfer coefficient in an exchanger are multiplicative faults. These introduce nonlinearities into residual generation and generally require more sophisticated detection methods [2].

By severity:

- **Hard faults**: Complete loss of function – a broken component, open circuit, or total sensor failure. Hard faults typically trigger immediate alarms and demand urgent response.
- **Soft faults**: Partial degradation that reduces performance but does not eliminate function entirely. Soft faults may progress to hard faults if left unaddressed, and early detection of soft faults is the primary value proposition of condition monitoring systems.

By origin:

- **Internal faults**: Faults originating within the system boundaries – component wear, manufacturing defects, material fatigue.
- **External faults**: Faults caused by environmental factors or operational conditions outside the system – overload, contamination, temperature extremes, electromagnetic interference.

The classification matters because diagnostic methods have different strengths for different fault types. Threshold-based detection catches abrupt hard faults easily but struggles with incipient soft faults. Model-based methods excel at detecting parametric (multiplicative) faults through parameter estimation but may be overly complex for simple additive sensor faults. Machine learning approaches can detect complex patterns across multiple fault types but require representative training data that includes the specific fault modes expected in operation.

A practical diagnostic system must account for the full spectrum of faults likely to occur in its application domain, not just the most dramatic failure modes. The costliest failures are often those that were possible but not anticipated – and therefore not monitored for.

Reliability Engineering Foundations

Diagnostics engineering cannot be separated from reliability engineering. Diagnostic systems exist to improve system reliability by enabling timely maintenance before catastrophic failure occurs. Understanding reliability metrics provides the quantitative framework for evaluating diagnostic effectiveness and justifying diagnostic system investment.

Failure rate and the bathtub curve: The failure rate, denoted by lambda (lambda), represents the instantaneous probability of failure at time t given that the system has survived to time t. For a population of similar components, the failure rate typically follows a characteristic pattern known as the bathtub curve, named for its U-shape [5].

The bathtub curve has three phases:

1. **Infant mortality (early life failures)**: A decreasing failure rate during the initial period of operation. Failures in this phase are caused by manufacturing defects, assembly errors, and material imperfections that cause early breakdown. Burn-in testing and factory acceptance tests aim to eliminate infant mortality failures before deployment.
2. **Useful life (random failures)**: A relatively constant failure rate during the main operational lifetime. Failures in this phase are random events caused by unpredictable stresses – overload events, environmental extremes, or latent defects that survive initial screening. The constant failure rate assumption underlies many reliability calculations and is valid for well-designed systems operating within specifications.
3. **Wear-out (aging failures)**: An increasing failure rate as components approach the end of their design life. Failures in this phase are caused by accumulated damage – fatigue crack growth, material wear, insulation degradation, lubricant breakdown. Wear-out failures are the primary target of condition monitoring and predictive maintenance systems.

The bathtub curve is a population-level model. Individual components do not cycle through all three phases; each component experiences one failure event (or reaches end-of-life without failure). The curve describes the aggregate behavior of many similar units over their operational lives.

Mean Time Between Failures (MTBF): For repairable systems, MTBF represents the average operating time between consecutive failures:

```
MTBF = Total Operating Time / Number of Failures
```

For a system with constant failure rate lambda during its useful life phase:

```
MTBF = 1 / lambda
```

MTBF is often misunderstood as a prediction of individual component lifetime. It is not. A system with MTBF of 10,000 hours does not fail every 10,000 hours; it means that across a large population, the average time between failures is 10,000 hours. Individual units may last 5,000 hours or 20,000 hours or longer.

Mean Time To Repair (MTTR): MTTR represents the average time required to restore system function after a failure. It includes detection time, diagnosis time, parts procurement, repair execution, and verification testing. Diagnostic systems directly reduce MTTR by enabling faster fault detection and more accurate fault isolation – a technician who knows exactly which bearing is failing can complete the repair much faster than one searching through multiple possible causes.

Availability: System availability quantifies the fraction of time a system is operational:

```
Availability = MTBF / (MTBF + MTTR)
```

For example, a system with MTBF of 10,000 hours and MTTR of 10 hours has availability of 10,000/10,010 = 99.9%. Improving either MTBF (through better design or predictive maintenance) or MTTR (through better diagnostics) improves availability.

Reliability function: The reliability R(t) is the probability that a component survives without failure to time t:

```
R(t) = e^(-lambda * t)    (for constant failure rate)
```

This exponential reliability function applies during the useful life phase with constant lambda. During wear-out, more complex distributions like Weibull or lognormal are used to model the increasing failure rate.

These reliability metrics connect directly to diagnostic system design. The target detection time for a diagnostic system depends on how quickly a fault progresses from incipient detection to catastrophic failure. The acceptable false alarm rate depends on the cost of unnecessary maintenance interventions versus the cost of missed faults. And the required diagnostic coverage – the fraction of possible fault modes that must be detected – depends on the safety criticality and availability requirements of the application.

The Cost of Failure: Risk Analysis and Consequence Modeling

Diagnostic systems are engineered investments, and like any investment, their design must balance cost against benefit. Understanding the economics of failure provides the framework for making rational diagnostic system design decisions.

The total cost of an unplanned failure includes direct costs (replacement parts, repair labor, emergency service premiums) and indirect costs (production downtime, lost revenue, contractual penalties, safety incidents, environmental damage, reputational harm). For critical industrial assets, indirect costs often exceed direct costs by factors of 10 to 100.

Consider a continuous chemical production process running at $50,000 per hour in revenue. An unplanned pump failure causes 8 hours of downtime for emergency repair, plus an additional 4 hours for restart and ramp-up. The total revenue loss is $600,000. If a condition monitoring system costing $25,000 to install and $5,000 per year to operate can predict such failures with sufficient lead time to schedule replacement during planned maintenance windows, the return on investment becomes obvious.

However, not all assets justify sophisticated diagnostic systems. A low-value component that is cheap to replace and quick to install may be better managed through run-to-failure strategy with spare parts inventory. The decision framework for diagnostic system investment considers:

- **Asset criticality**: How important is the asset to overall system function?
- **Failure consequence severity**: What are the safety, environmental, and economic impacts of failure?

- **Failure detectability**: Can the fault be detected before catastrophic failure using available sensor technology?
- **Degradation timescale**: Does the fault develop slowly enough to enable planned intervention, or is it abrupt with no warning?
- **Maintenance accessibility**: How easy is it to access and repair the component when a fault is detected?

Risk matrices that plot consequence severity against failure probability provide a structured framework for prioritizing diagnostic system investment. Assets in the high-consequence, high-probability quadrant demand the most sophisticated diagnostic capability. Assets in the low-consequence, low-probability quadrant may need minimal or no dedicated diagnostics.

Design Patterns in Diagnostic Systems

Diagnostic systems follow recurring architectural patterns that have evolved through decades of practice across industries. Understanding these patterns provides a vocabulary for discussing diagnostic system design and a starting point for adapting proven architectures to new applications.

Centralized monitoring architecture: All sensor data is collected and transmitted to a central processing unit where diagnostic algorithms run. This pattern is common in industrial condition monitoring systems where vibration, temperature, and acoustic sensors on multiple machines feed into a central server running analysis software. Advantages include centralized expertise, shared computational resources, and unified alarm management. Disadvantages include single points of failure, network dependency, and latency for time-critical diagnostics.

Distributed (edge) diagnostic architecture: Diagnostic processing occurs at or near the sensor, with only summary information (health indices, alarm states, feature vectors) transmitted to central systems. This pattern is increasingly common in IoT deployments where thousands of sensors generate data that would overwhelm network bandwidth if transmitted raw. Advantages include reduced bandwidth requirements, lower latency for local decisions, and resilience against network failures. Disadvantages include limited computational resources at the edge, difficulty updating diagnostic algorithms across distributed nodes, and fragmented data that complicates fleet-level analysis.

Hierarchical diagnostic architecture: Multiple levels of diagnostic processing with increasing sophistication at higher levels. Edge sensors perform simple threshold checks and feature extraction. Local controllers aggregate features from multiple sensors and run classification algorithms. Central systems perform trend analysis, prognosis, and maintenance optimization across the entire asset fleet. This pattern balances the strengths of centralized and distributed approaches and is common in large-scale industrial deployments [6].

Observer-based diagnostic architecture: A mathematical model of the system runs in parallel with the physical system, continuously comparing model predictions against actual measurements to generate residuals. This pattern is fundamental to model-based diagnostics and appears in aerospace flight control systems, automotive engine management, and industrial process control. The observer (whether a Luenberger observer, Kalman filter, or neural network surrogate model) acts as a virtual sensor that provides redundancy for physical sensors.

Data-driven diagnostic architecture: Machine learning models trained on historical data perform classification and anomaly detection directly on sensor inputs without explicit physical modeling. This pattern is increasingly common in applications where physical models are difficult to develop (complex nonlinear systems, multi-physics interactions) or where vast amounts of labeled training data are available. The key challenge is ensuring model generalization across operating conditions not represented in the training data.

These patterns are not mutually exclusive. Modern diagnostic systems often combine multiple patterns – for example, using observer-based residual generation for model-based detection while simultaneously running a machine learning classifier on the same sensor data, with results fused at a higher decision level. The choice of architecture depends on application requirements for response time, computational resources, communication infrastructure, and the nature of faults to be detected.

Chapter 2: System Modeling for Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

First-Principles Physical Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

System Identification from Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Bond Graphs and Energy-Based Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Model Validation and Fidelity Assessment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Digital Twins: Bridging Physical and Virtual Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Modeling a DC Motor Drive System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 3: Sensor Technologies and Signal Acquisition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Sensor Fundamentals: Transduction Mechanisms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Signal Conditioning: Amplification, Filtering, and Noise Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Analog-to-Digital Conversion: Sampling Theory and Aliasing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Sensor Selection Criteria and Trade-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Sensor Selection and DAQ Design Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Data Acquisition System Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Vibration Monitoring Sensor Chain Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 4: Signal Processing for Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Time-Domain Features and Statistical Indicators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Fourier Analysis and Spectral Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Wavelet Transforms and Time-Frequency Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Order Tracking and Synchronous Averaging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Envelope Analysis and Demodulation Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Extracting Bearing Fault Frequencies from Vibration Data

FFT Algorithm: Cooley-Tukey Radix-2 Implementation

Envelope Analysis Pipeline: Complete Implementation

Bearing Defect Frequency Derivation

Signal Processing Method Selection Checklist

Chapter 5: Model-Based Fault Detection and Diagnosis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Residual Generation: The Core Mechanism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Parity Space Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Observer-Based Methods: Luenberger Observers and Kalman Filters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Parameter Estimation for Fault Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Robustness vs. Sensitivity: The Fundamental Trade-off in Residual Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Residual Generator Design Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Actuator Fault Detection in an Aircraft Control Surface

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 6: Data-Driven and Machine Learning Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

The Data-Driven Paradigm: When Models Fail but Data Abounds

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Supervised Learning for Fault Classification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Unsupervised Anomaly Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Deep Learning for Diagnostics: CNNs, RNNs, and Transformers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Training Data Challenges: Imbalanced Classes, Label Noise, and Domain Shift

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Machine Learning Model Development and Deployment Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Machine Learning-Based Gearbox Fault Classification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 7: Condition Monitoring and Health Management Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Condition Monitoring System Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Threshold Design: Statistical vs. Model-Based Approaches

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Alarm Management and False Alarm Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Health Indices and Degradation Tracking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Prognostics and Remaining Useful Life Estimation (RUL Prediction)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Turbomachinery Health Monitoring System Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 8: Embedded Diagnostics and Real-Time Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Embedded Diagnostic Architecture Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Real-Time Constraints: Latency, Throughput, and Determinism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Resource-Constrained Algorithms: Fixed-Point Arithmetic and Memory Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Built-In Self-Test (BIST) for Digital Circuits and Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Hardware-Level Diagnostics: Watchdog Timers, CRC, and ECC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: ECU Onboard Diagnostics Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 9: Automotive Diagnostics Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

OBD-II Standards and Regulatory Requirements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Diagnostic Trouble Codes (DTCs): Structure, Classification, and Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Unified Diagnostic Services (UDS) Protocol: ISO 14229

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

CAN Bus Diagnostics and Network Health Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Powertrain Diagnostics: Engine, Transmission, and Aftertreatment Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Automotive Diagnostics Compliance Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: OBD-II Monitor Design for Catalytic Converter Efficiency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 10: Industrial Diagnostics and Predictive Maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Industrial Condition Monitoring Standards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Rotating Machinery Diagnostics: Motors, Pumps, Compressors, Turbines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Process Industry Diagnostics: Valve Stiction, Heat Exchanger Fouling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Predictive Maintenance Strategies and ROI Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Integration with Enterprise Systems: MES, ERP, and CMMS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Industrial Condition Monitoring System Deployment Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Motor Condition Monitoring System for a Manufacturing Plant

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 11: Software Diagnostics and Fault-Tolerant Computing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Software Fault Detection Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Fault Tolerance Patterns: Redundancy, Checkpointing, Recovery Blocks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Memory Diagnostics: ECC, Scrubbing, and Error Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Communication Protocol Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Runtime Verification and Formal Methods for Safety-Critical Software

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Software Fault Tolerance Design Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Fault-Tolerant Controller Design for an Autopilot System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 12: Validation, Verification, and Testing Methodologies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Diagnostic Performance Metrics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Fault Injection Testing: Methods and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Test Data Generation: Real vs. Simulated Fault Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

ROC Curves and Decision Threshold Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Standards and Certification: DO-178C, ISO 26262, IEC 61508

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Diagnostic Validation Testing Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Worked Example: Validating a Vibration-Based Bearing Diagnostic System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Chapter 13: Emerging Trends and Future Directions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

AI-Assisted Diagnostics: From Assisted Analysis to Autonomous Diagnosis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Digital Twins at Scale: Fleet-Wide Health Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Edge Computing and Distributed Diagnostic Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Federated Learning for Privacy-Preserving Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Quantum Sensing and Next-Generation Sensor Technologies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

The Future of Diagnostics: Convergence and Open Challenges

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Emerging Technology Adoption Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/diagnosticsengineering.