A Discrete Event Systems Approach to Failure Diagnosis: Theory & Applications

Stéphane Lafortune
Department of EECS, University of Michigan
&
Meera Sampath
Joseph C. Wilson Center for Research & Technology
Xerox Corporation

DX-00 -Eleventh International Workshop on Principles of Diagnosis
2000/06/08
Diagnostics in the Industrial World

• The Three C’s:
  Cost, Computation, and Customer Satisfaction

  - Downtime is unproductive and undesirable.
  - Service is costly and competitive.

• Safety
• Health Regulations
Requirements for Industrial Systems

• Diagnostic engine must be easy to develop.

• Diagnostic engine must be simple to implement.

• Diagnosis must be achieved with minimal, cost-effective set of sensors.

• Diagnosis may need to be achieved with decentralized information
The “Academic” Viewpoint

Automated Diagnostic Methodologies that:

- Are formal and model-based
- Are applicable to dynamic systems
- Allow analysis of diagnosability properties
- Are “easy” to implement
- Are extensible and versatile
The “DES” Diagnostic Methodology

DES: Discrete-Event Systems

- Modeling: languages and automata
- Dynamic tracking and state-based inferencing: Diagnosers
- Ability to incorporate sensor information from multiple sources: real and virtual sensors
- Automated design of diagnostic inference engine
- Simple on-line implementation
Conceptual View for Automated Systems

SUPERVISORY CONTROLLER

REAL-TIME CONTROL

DIAGNOSTICS

FAILURE RECOVERY

INTERFACE

SYSTEM CONTROLLERS

SYSTEM SENSORS

SYSTEM
Implementation
Motivating Example 1: Heating, Ventilation, and Air Conditioning Systems

- Components hard to access, few sensors
- Valve, pump, controller faults, etc.
- Sinnamohideen, Sampath et al., JCI
Motivating Example 2: Document Processing Systems

- Complex processes, few sensors
- Electro-mechanical and image quality faults
- Sampath et al., Xerox Corp.
Motivating Example 3: Automated Highway Systems (AHS)

- Platoons of vehicles
- Transmitter and receiver faults
- Sengupta et al., PATH, UC-Berkeley
Contents of Presentation

• “Basic” theory
  – Notion of diagnosability
  – Model construction
  – Diagnosers: synthesis and analysis

• Industrial applications
  – “Hybrid” techniques for Document Processing Systems

• Decentralized architectures
  – Coordinated architectures
  – Performance – Complexity tradeoff
Diagnosability: The Premise

- Event-based model: traces of events
- Language: set of traces of events
- Fault or failure events: unobservable
- Partition failures into types
Diagnosability: Intuitive Statement

A language (i.e., DES) is **diagnosable** with respect to a partition of the failure events and with respect to a set of observable events if it is possible to detect occurrences of any type of failure with finite delay, based on observed event sequences only.
• trace $s$ ends with unobservable failure event $f_i$
• trace $t$ is a sufficiently long continuation of trace $s$
• “any trace of the system that looks like $st$ must contain a failure event of same type as $f_i$”
• multiple failures may occur
Definition of Diagnosability

A prefix-closed and live language $L$ is said to be \textit{diagnosable} with respect to the projection $P$ and with respect to the partition $\Pi_f$ on $E_f$ if the following holds:

\[
( \forall i \in \Pi_f ) ( \exists n_i \in \mathbb{N} ) ( \forall s \in \Psi(E_{fi}) ) \\
( \forall t \in L/s ) [ || t || \geq n_i \Rightarrow D ]
\]

where the diagnosability condition $D$ is:

\[
\omega \in P_{L}^{-1} [P(st)] \Rightarrow E_{fi} \in \omega.
\]
Explanation of Notation

• Projection $P$ : “erases” unobservable events
  
  $$P(\ o1\ uo1\ uo2\ o2\ uo3\ o3\ ) = o1\ o2\ o3$$

• Inverse Projection $P_L^{-1}$ :
  
  $$P_L^{-1}(y) = \{ \ s \in L : P(s) = y \}$$

• Traces ending in failure of type $i$ :
  
  $$\Psi(E_{fi}) = \{ \ sa \in L : a \in E_{fi} \}$$
Diagnosability: Illustrative Example

- If \( F_1 = \{f_a\} \), \( F_2 = \{f_b\} \), and \( F_3 = \{f_c\} \), then not diagnosable
- If \( F_1 = \{f_a, f_b\} \) and \( F_2 = \{f_c\} \), then diagnosable
Why a Formal Notion of Diagnosability?

- Analysis:
  - Is this DES diagnosable?

- Design:
  - What sensors to use
  - How to use them
  - What changes to make to the system
Steps in the DES Approach

1. **Step 1: Build Discrete Event Model of System**
   - Component Models
   - Test Sequence/Controller Model
   - Fault Symptom Tables
   - System Model

2. **Step 2: Build Diagnoser**
   - Diagnoser
   - Diagnostic Requirements

3. **Analysis:** Is it Diagnosable?
4. **Design:** How to Diagnose?
Building Discrete-Event Model of System

- Use automata (or state machines) as basic building blocks to model components
- Obtain the parallel composition of components
- Incorporate the sensor information in the event set
- Obtain the complete model as an automaton
Simple Pump-Valve-Controller Example

- OPEN_VALVE, CLOSE_VALVE
- STOP_PUMP
- START_PUMP
- STUCK_CLOSED
- STUCK_OPEN
- OPEN_VALVE
- CLOSE_VALVE
- C1
- C2
- C3
- C4

- VC
- VO
- SC
- SO

- POFF
- PON

- OPEN_VALVE
- CLOSE_VALVE

- Stéphane Lafortune, Dept. of EECS
- Meera Sampath, Wilson Center for Research & Technology
The Global Sensor Map

<table>
<thead>
<tr>
<th>STATES</th>
<th>SENSOR MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-6, 10-12: (POFF, -, -)</td>
<td>NP, NF</td>
</tr>
<tr>
<td>7:</td>
<td>(PON, VO,C3)</td>
</tr>
<tr>
<td>8:</td>
<td>(PON, SO,C3)</td>
</tr>
<tr>
<td>9:</td>
<td>(PON, SC,C3)</td>
</tr>
</tbody>
</table>
The Event Set of the Composite Model

• **Observable events:**
  – Commands issued by controller (with sensor readings)
  – Changes in sensor readings
  – “Generalized events”: changes in virtual sensor readings, test outcomes, etc.

• **Unobservable events:**
  – Failures of components, controllers, sensors
  – Changes in system state not recorded by sensors
(Part of) HVAC System

- FAN
- HTG. COIL
- VALVE
- PUMP
- BOILER
- CONTROLLER
Steps in the DES Approach

**Step 1:** Build Discrete Event Model of System  **Step 2:** Build Diagnoser

- Component Models
- Test Sequence/Controller Model
- Fault Symptom Tables
- System Model
- Diagnoser
- Diagnostic Requirements

**Analysis:** Is it Diagnosable?

**Design:** How to Diagnose?
Tool for Analysis and On-line Diagnosis

- Let the composite model generate language $L$
- Pick any automaton $G$ that generates $L$
- The diagnoser $G_d$ is an automaton built from $G$
  - Think of $G_d$ as a “refined” observer:
    - $G_d$ carries state estimates
    - $G_d$ carries failure labels
Illustrative Example: $G_d$

- **F1**: SC
- **F2**: SO

Diagram showing states and transitions with actions like `OPEN_VALVE`, `START_PUMP`, `CLOSE_VALVE`, and input states `F1` and `F2`.
Meera Sampath, Wilson Center for Research & Technology

SC
<OPEN_VALVE, NP,NF>
<OPEN_VALVE, NP,NF>
<OPEN_VALVE, NP,NF>
<CLOSE_VALVE, NP,NF>
<CLOSE_VALVE, NP,NF>
<STOP_PUMP,NP,NF>
<STOP_PUMP,NP,NF>
<STOP_PUMP,NP,NF>
<STOP_PUMP,NP,NF>
< START_PUMP,PP,F >
< START_PUMP,PP,F >
< START_PUMP,PP,F >
< START_PUMP,PP,F >
< F -> NF >
Construction of the Diagnoser

- States are of the form: \{(x_1, l_1), \ldots, (x_n, l_n)\}
  - \(x_i\) are states of system G
  - \(l_i\) are labels: N or subsets of \{F_1, \ldots, F_n\}
- Transitions are due to observable events only
- Update of state estimates: similar to conversion of nondeterministic automaton to deterministic one
- Failure labels are propagated and updated by failures encountered along unobservable subtraces
State Transition Function of Diagnoser

F1: SC
F2: SO
Diagnoser $G_d$

F1: SC
F2: SO
Information in Diagnoser States

Uncertain for:
F1, F2, F3, F4

Uncertain for: F1, F2

Certain for F3

(from HVAC example)
What Should We Worry About?

F1: SC
F2: SO
Formal Result

• The language $L$ is diagnosable iff the diagnoser $G_d$ does not contain any indeterminate cycles

• Indeterminate cycles in $G_d$ are cycles of uncertain states that have corresponding cycles in $G$ involving their failed states

• This necessary and sufficient condition is implementable (polynomial complexity)
Example of Indeterminate Cycle

<START_PUMP,PP,F>

<STOP_PUMP,NP,NF>

<OPEN_VALVE, NP,NF>

<CLOSE_VALVE, NP,NF>

<OPEN_VALVE, NP,NF>

<STOP_PUMP,NP,NF>

<STOP_PUMP,NP,NF>
The Notion of Indeterminate Cycle

• Intuition:

An indeterminate cycle corresponds to the situation where there are two traces in L, of arbitrary long length, that have the same observable projection, and where

- one trace contains a failure event of a certain type
- the other trace does not
The Notion of Indeterminate Cycle

• Formally:

An **Fi-indeterminate** cycle in $G_d$ is a cycle of Fi-uncertain states for which there exists:
  → a corresponding cycle (of observable events) in $G$ involving only states that carry Fi in their labels in the cycle in $G_d$
Not all Uncertain Cycles are Indeterminate!
Steps in the DES Approach

1. Build Discrete Event Model of System
2. Build Diagnoser

Analysis: Is it Diagnosable?
Design: How to Diagnose?

Component Models
Test Sequence/Controller Model
Fault Symptom Tables
System Model
Diagnoser
Diagnostic Requirements
On-Line Diagnosis Using Diagnosers

• Store transition function of diagnoser
• Update state after each observable event
• Report failure status based on labels in state
• Formal Result:
  • If a given system (language) is diagnosable, then the diagnoser detects occurrences of failure events of any type in a bounded number of events after the occurrence of the failure event
  • “Detects” ≡ Enters a Certain State
Steps in the DES Approach: Case of “On-the-Fly” Computations

Component Models

Test Sequence/Controller Model

Fault Symptom Tables

On-the-fly calculation of diagnoser state

Diagnostic Requirements

Diagnostic decision
On-Line Diagnosis Using Diagnosers: Case of “On-the-Fly” Computations

- Store component models and sensor tables
- Calculate current diagnoser state after each observable event:
  - Using component models and sensor tables, build current state of system plus some limited lookahead (until next observable event)
  - Build current diagnoser state
How to Achieve Diagnosability?

Step 1: Build Discrete Event Model of System
Step 2: Build Diagnoser

Analysis: Is it Diagnosable?

Design: How to make system diagnosable?
How to Make System Diagnosable?

• Select **new set of sensors** (i.e., observable events), and repeat process of building and testing diagnoser

• Problem of optimal sensor selection:
  - Given set $A$ of available sensors, select **minimum-cost** subset of $A$ for which system is diagnosable
    → Need efficient testing strategy
    → See *Debouk et al.*, CDC 99
How to Make System Diagnosable?

• Integrate *supervisory control* and failure diagnosis:
  • Design a control protocol that makes system diagnosable *and* that achieves control objectives
  • Different control protocols lead to different diagnosability properties!
Modified Pump-Valve-Controller Example

- OPEN_VALVE, CLOSE_VALVE
- STOP_PUMP
- START_PUMP
- STUCK_OPEN
- STUCK_CLOSED
- OPEN_VALVE
- CLOSE_VALVE
Active Diagnosis Problem

- Need results from supervisory control theory
- See Sampath et al., CDC 97 and TAC 98
How to Make System Diagnosable?

- Do more “intelligent” processing of available information, using complementary diagnostic techniques
  - Concept of virtual sensor
    
    *Sampath et al., Xerox Corp.*
Contents of Presentation

• “Basic” theory
  – Notion of diagnosability
  – Model construction
  – Diagnosers: synthesis and analysis

• Industrial applications
  – “Hybrid” techniques for Document Processing Systems

• Decentralized architectures
  – Coordinated architectures
  – Performance – Complexity tradeoff
Industrial Applications: Hybrid Techniques for Document Processing Systems
Requirements for Industrial Systems

- Diagnostic engine must be easy to develop.

- Diagnostic engine must be simple to implement.

- Diagnosis must be achieved with minimal, cost-effective set of sensors.

- Diagnosis may need to be achieved with decentralized information
A Hybrid Approach to Failure Diagnosis

Qualitative Model based Diagnostic Engine
+
Quantitative Analysis based “Virtual Sensors”

SYSTEM
VIRTUAL SENSORS
SYSTEM SENSORS
SYSTEM CONTROLLER(S)
EVENT GENERATOR
QUALITATIVE MODEL
DIAGNOSTIC INFEREN CE ENGINE
FAILURE OUTPUT
VIRTUAL SENSORS
SYSTEM
SYSTEM SENSORS
SYSTEM CONTROLLER(S)
Virtual Sensor Examples

• Signal Processing
  Signature Analysis, Spectrum Analysis, Vibration Analysis

• Statistical Analysis
  Means, Variance, Whiteness, Clustering

• State Estimation Techniques
  Kalman Filter

• Qualitative Techniques
  Qualitative Calculus, Constraint Propagation
Other Hybrid Approaches to Diagnosis

- Frank, P.M. - Expert Systems & FDI Schemes (1990)
- Pomeroy et. al. - Model based Diagnosis & FDI Schemes (1990)
- McIlraith et al. - Model based Diagnosis & Parameter Estimation (1999)
- Zhao et. al - Model based Diagnosis & Signal Processing (2000)
Two Examples from the Document Processing Industry

- Embedded Diagnostics of the Paper Feeder System in a Digital Copier
  - Signature Analysis based Virtual Sensor

- A System for Automated Diagnosis of Image Quality Problems
  - Image Processing based Virtual Sensor
Application 1: Real Time Diagnosis of the Paper Feeder System in a Digital Copier

The Xerox Document Center DC265
The DC265 Paper Feeder System Components

- Paper Trays
- Wait Station Sensor
- Drives Plate - Feed & Elevator Motors, Paper Size Sensors
- Feed Roll Cartridge
- Stack Height Sensor
- Nudger Solenoid,
Paper Feeder Diagnostic System

DIGITAL DATA STREAM

DISCRIMINANT ANALYSIS

FEATURE EXTRACTION

ANALOG DATA STREAM

EVENT GENERATOR

DIAGNOSER

LOCAL UI

REMOTE UI

Failure 2

Cluster 2

Event 1: Feed_Motor_On
Event 5: Stack Ht. Counter Low

FAILURE

Broken Feed Roll Cartridge

Failure 2

Paper Feeder Diagnostic System
Signature Analysis & Feature Extraction based Virtual Sensor

- Drives Plate ground current is a “good” diagnostic indicator
- Feature extraction based analysis computationally less expensive
- Choice of Features for Paper Feeder Assembly:
  - Peak Current
  - Power Spectral amplitudes
- Feature Extraction followed by Statistical Discriminant Analysis
- Choice of Classifiers:
  - Linear Classifiers
  - Quadratic classifiers
Discriminant Analysis - Sample Results

Quadratic Classifier: 

\[ d_i^2(x) = (x - m_i)'(S_i^{-1})(x - m_i) + \ln |S_i| \]

\( X_i \) – Sample Observation; \( M_i \) – Mean of Cluster \( i \); \( S_i \) – Covariance of Cluster \( i \)
DIAGNOSER DESIGN

Analysis: Is it Diagnosable?

Design: How to Diagnose?

Step 1: Build Discrete Event Model of System

Step 2: Build Diagnoser

SENSOR MAPS & VIRTUAL SENSOR OUTPUTS

Component Models

Test Sequence/Controller Model

Fault Symptom Tables

System Model

Diagnoser

Diagnostic Requirements
(Part of The) Diagnoser for the Paper Feeder System

- **Start Feed Cycle**: N F1 F2 F3 F4 F5 F6
- **Feed Motor On**: N F1 F2 F3 F4 F5 F6
- **Solenoid On**: N F1 F2 F3 F4 F5 F6
- **Feed Motor Off, Wait Sensor Low**: F1 F2 F3 F4 F6
- **Peak Current High**: F1 F2 F3 F4 F6
- **Stack Height Counter High**: F1 F3 F6

**F1 - Out Of Paper**
**F2 - Stalled Feed Motor**
**F3 - Stalled Elev Motor**
**F4 - Broken Solenoid**
**F5 - Degraded Solenoid**
**F6 - Broken Feed CRU**
Experimental Setup
Diagnostic System Output
Key Concepts Addressed

- Real-Time Embedded Diagnosis
- Diagnosis with Minimal Sensor Requirements
- Component/FRU/CRU Level Diagnosis
- Predictive Diagnosis
- Customer Repair
- Remote Notification
Application 2: Automated Image Quality Diagnostics System

- An automated diagnostic system to resolve IQ related problems in printing systems

- A tool for the customer

- Designed to diagnose machine failures resulting in common IQ problems such as Banding, Mottle, Spots, etc.

- IQ Defect Diagnosis non-trivial due to many-one failure mapping
Image Quality Diagnostics System

- User Interface
- Diagnostic Inference Engine
- Repair Planning
- Event Generator
- Xerographic Tests
- Machine Interface
- Feature Extraction
- IQ Defect Analysis
- Scanner Interface
- Test Pattern

This is a test pattern
Image Quality Diagnostics System

- Image Processing based Virtual Sensor

- The IQAF tool suite for Image Quality Analysis developed at the Wilson Center for Research at Xerox

- Techniques include Filtering, Transformations, FFTs, Statistical Analysis

- “On-Demand” Virtual Sensors

- Integrated DES / Bayesian Analysis based diagnostic engine
Image Quality Diagnostics - Prototype System

- Prototype System on the Xerox DC265 with UMax Scanner
- Image Quality Defect - Bands & Streaks
- 10 Failures - Xerographic Engine & Optics
  - Degradation, Wear & Tear, Contamination
Image Quality Defect Examples

A

B
Image Quality Analysis Outputs

IQ Virtual Sensor Outputs: Defect Presence, Defect Orientation, Defect Polarity, Defect Spread, etc.
(Part of the) Diagnostic Inference Engine

- **Uniformity Test, Pass**
  - N 0.04  F1 0.16  F2 0.04  F3 0.24  F4 0.16  F5 0.04  F6 0.04  F7 0.04  F8 0.24

- **Charge Test, Pass**
  - N 0.04  F1 0.09  F2 0.04  F3 0.26  F4 0.17  F5 0.04  F6 0.04  F7 0.04  F8 0.26

- **Defect Presence, All Pages**
  - N 0.05  F1 0.12  F2 0.03  F3 0.19  F4 0.12  F5 0.03  F6 0.05  F7 0.05  F8 0.33

- **Defect Type, Streak**
  - N 0.00  F1 0.21  F2 0.00  F3 0.31  F4 0.21  F5 0.05  F6 0.09  F7 0.09  F8 0.01

- **Defect Spread, Uniform**
  - N 0.04  F1 0.21  F2 0.00  F3 0.31  F4 0.21  F5 0.05  F6 0.09  F7 0.09  F8 0.01

**F3 - Failed Charge Corotron**
Highlights of The Hybrid Diagnostic Scheme

- A unified framework to integrate a variety of diagnostic techniques
- Generalized notion of an event
- A hybrid extension that retains the analytical properties of the DES methodology
- A diagnostic technique motivated by industrial considerations
Contents of Presentation

• “Basic” theory
  – Notion of diagnosability
  – Model construction
  – Diagnosers: synthesis and analysis

• Industrial applications
  – “Hybrid” techniques for Document Processing Systems

• Decentralized architectures
  – Coordinated architectures
  – Performance – Complexity tradeoff
Motivating Example 3: Automated Highway Systems (AHS)

- Platoons of vehicles
- Transmitter and receiver faults
- Sengupta et al., PATH, UC-Berkeley
AHS: Platoon Communication

- Wireless LAN, TDMA, 20 msec, for velocity and acceleration data
- Separate (reliable) communication channel to exchange diagnostic information about LAN
- Model considered: leader and two followers
Partial system model

\[ F_3 = f_1r_f \]

\[ F_1 = l_tf \]

l: leader
fi: follower i
Issue:

• LAN faults cannot be diagnosed “individually” (namely, by running three independent diagnosers)

• A centralized diagnostic scheme is not practical

Questions:

• Can the vehicles “jointly” diagnose the LAN faults, by sharing “some” information

• How to proceed?  ➔ Sengupta et al.
  ➔ Debouk et al.
Decentralized Diagnosis with Coordinator

COORDINATOR → FAILURE RECOVERY

Communications

SUPERVISORY CONTROLLER 1 → DIAGNOSTICS

Local Observations

INTERFACE 1

SUPERVISORY CONTROLLER 2 → DIAGNOSTICS

INTERFACE 2

SITE 1

SITE 2
Key Ingredients

- Local processing for diagnostics
- Communication rule
- Decision rule at coordinator

We call these a PROTOCOL
Diagnosability in this New Architecture

A language (i.e., DES) is diagnosable with respect to a protocol, a partition of the failure events, and sets of locally observable events if under this protocol, the coordinator site can detect the occurrence of any type of failure with finite delay.
Modified Definition of Diagnosability

A prefix-closed and live language $L$ is said to be diagnosable with respect to the given protocol, the projections $P_1$ and $P_2$, and the partition $\Pi_f$ on $E_f$ if the following holds:

$$\forall i \in \Pi_f \ (\exists n_i \in N) \ (\forall s \in \Psi(E_{fi}))$$

$$\forall t \in L/s \ [\ |t| \geq n_i \Rightarrow C \text{ is Fi-certain}]$$

C: register holding diagnostic information at the coordinator site
Key Assumptions

- System is diagnosable in a centralized set up
- One site alone cannot diagnose all faults
- Communication is reliable:
  - Global ordering is preserved at the coordinator
  - No raw data is communicated
  - Communication may be interrupted
- Coordinator should be “simple”
  memory, processing, no system model
Objective

- Design a set of protocols and analyze their “complexity – performance” tradeoff
- Compare their performance to the centralized diagnoser

The centralized scheme is the “only” one available for comparison purposes…
Work Done So Far

- Protocol 1
  - Performance decreases
- Protocol 2
  - Complexity increases
- Protocol 3
Brief Description of Protocol 3

- Diagnosers are used at local sites
- Communicate nothing but failures detected (Fi-certain)
- Coordinator is “trivial”
- Test to determine if protocol works
Brief Description of Protocol 2

- Diagnosers are used at local sites
- Communicate current diagnoser state to coordinator
  - Communicate after each observable event
  - May interrupt communication
- Do simple “intersections” at coordinator
Protocol 2: A Few More Details

- Communicate also:
  - status bit (common event or not)
  - the unobservable reach
- Coordinator only stores most recent message from each site
Protocol 2: Decision Rule:
Upon Reception of Message $R_1$ from Site 1

<table>
<thead>
<tr>
<th>Rule</th>
<th>Current Status Bit</th>
<th>Status Bit Received</th>
<th>Coordinator</th>
<th>Status Bit Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Rule 1</td>
<td>0</td>
<td>0</td>
<td>$R_1 \cap R_4$</td>
<td>0</td>
</tr>
<tr>
<td>Decision Rule 3</td>
<td>0</td>
<td>1</td>
<td>Wait</td>
<td>1</td>
</tr>
<tr>
<td>Decision Rule 5</td>
<td>1</td>
<td>1</td>
<td>$R_1 \cap R_2$</td>
<td>0</td>
</tr>
</tbody>
</table>
Protocol 2: Decision Rule:

Upon Reception of Message $R_1$ from Site 1

$R_2$: last message from site 2 w/out unobservable reach

$R_4$: last message from site 2 with unobservable reach
AHS: Platoon Communication

Protocol 3 does not work
Protocol 2 works!
## AHS: Platoon Communication – Protocol 2

Trace: \textit{ltf} tick tick tick tick tick

<table>
<thead>
<tr>
<th>Event</th>
<th>Lead</th>
<th>Foll 1</th>
<th>Foll 2</th>
<th>Coord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>1N</td>
<td>1N</td>
<td>1N</td>
<td>1N</td>
</tr>
<tr>
<td>\textit{ltf}</td>
<td>1N</td>
<td>1N</td>
<td>1N</td>
<td>1N</td>
</tr>
<tr>
<td>tick</td>
<td>F1,\ldots,F6 uncertain</td>
<td>F1,F3 uncertain</td>
<td>F1,F6 uncertain</td>
<td>F1 certain</td>
</tr>
<tr>
<td>tick</td>
<td>F1,\ldots,F6 uncertain</td>
<td>F1,F3 uncertain</td>
<td>F1,F6 uncertain</td>
<td>F1 certain</td>
</tr>
<tr>
<td>tick</td>
<td>F1,\ldots,F4 uncertain</td>
<td>F1,F3 uncertain</td>
<td>F1,F6 uncertain</td>
<td>F1 certain</td>
</tr>
<tr>
<td>tick</td>
<td>F1,F2 uncertain</td>
<td>F1,F3 uncertain</td>
<td>F1,F6 uncertain</td>
<td>F1 certain</td>
</tr>
</tbody>
</table>
Analytical Results

• Protocol 2 does not always work!
• Test available to verify if it will detect all failures
• Sufficient condition available
  → failure-ambiguous traces

• Communication can be interrupted: coordinator could do polling
Example where Protocol 2 fails

\[ \Sigma_1 = \{ a, c, d, e \} \]
\[ \Sigma_2 = \{ b, d, e \} \]
\[ \Sigma_{ni} = \{ \sigma_1 \} \]
Summary

Uses *extended* diagnosers and 1-step memory at coordinator

Diagnosers + state intersection

Diagnosers but trivial/no coordinator

- Protocol 1: Performance decreases
- Protocol 2: Complexity increases
- Protocol 3: Diagnosers but trivial/no coordinator
Decentralized Diagnosis with Coordinator

- **COORDINATOR**
  - **FAILURE RECOVERY**
  - Communications
  - Local Observations

- **SUPERVISORY CONTROLLER 1**
  - **DIAGNOSTICS**
  - **INTERFACE 1**
  - **SITE 1**

- **SUPERVISORY CONTROLLER 2**
  - **DIAGNOSTICS**
  - **INTERFACE 2**
  - **SITE 2**

- **INTERFACE 1**
  - **INTERFACE 2**
Communication Delays

• **Time stamps** appended to messages

• “**Ordering**” of messages at coordinator:
  
  • Store messages (need upper bound on delay)
  
  • Sort out all possible orders
  
  • Apply earlier decision rule to each order

  → Requires more memory and processing at coordinator…
Contents of Presentation

• “Basic” theory
  – Notion of diagnosability
  – Model construction
  – Diagnosers: synthesis and analysis
• Industrial applications
  – “Hybrid” techniques for Document Processing Systems
• Decentralized architectures
  – Coordinated architectures
  – Performance – complexity tradeoff
• Concluding remarks
Salient Features of Approach

• Formal: model-based, diagnosability
• Applicable to dynamic systems
• Analytical foundations:
  • diagnosers, indeterminate cycles, failure-ambiguous traces
• Amenable to design:
  • sensor selection, active diagnosis
• Easy of implementation
• Extensible, versatile
Other Extensions of “Basic” Theory

• Timed models of DES
  – Chen & Provan, Rockwell, ACC 97
  – Zad et al., Univ. of Toronto, CDC 99 (see also CDC 98)

• Decentralized DES
  – Sengupta et al., PATH–U.C. Berkeley, WODES 98
  – Rozé and Cordier, IRISA, WODES 98
  – Pencolé, IRISA, DX-00

• Modular DES
  – Ricker et al., IRISA
Other DES Approaches to Diagnostics

- Bouloutas et al., Columbia, 1992
  [automata, telecommunication networks]
- Lin, WSU, 1994
  [automata, automotive]
- Holloway et al., Kentucky, 1994 - present
  [time templates, manufacturing]
- Benveniste et al., IRISA, 1997 - present
  [stochastic Petri nets and automata, telecommunication networks]
- Baroni et al., Brescia, Italy, 1999
  [communicating automata]
Challenges Ahead

- Large-scale systems:
  - Decoupling, modularity
- Decentralized-information systems
  - Novel architectures
- Imprecise information
  - Probabilistic extension
- More industrial applications
Acknowledgements

Xerox Corporation
- Charles Coleman
- Ashok Godambe
- Eric Jackson
- Edward Mallow
- Rene Rasmussen
- Ronald Rockwell

University of Michigan
- Rami Debouk
- Demosthenis Teneketzis
- UMDES-LIB Group

University of California at Berkeley
- Raja Sengupta

Rockwell Science Center
- Yi-Liang Chen
- Gregory Provan

Johnson Controls Inc.
- Kasim Sinnamohideen
THE END

FIN
Some Words of Wisdom

Taoism says: S... happens.

Hinduism says: This s... happened before.

Buddhism says: If s... happens, it isn't really s...