

Automating Root-Cause Analysis: EMC Ionix Codebook Correlation Technology vs. Rules-based Analysis

Technology Concepts and Business Considerations

Abstract

This white paper discusses the role of root-cause analysis (RCA) in managing infrastructure availability and performance. By comparing two approaches – rules-based analysis and the EMC[®] Ionix[™] Codebook Correlation Technology[™] (CCT) – this paper illustrates the unique advantages of CCT and explains why CCT is the only viable method for automating the analysis of any type of problem in any networked system, no matter how complex. It also demonstrates how CCT provides uniquely powerful development- and maintenance-free intelligent analysis that automatically adapts to the managed environment.

November 2009

Copyright © 2001, 2005, 2009 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com

All other trademarks used herein are the property of their respective owners.

Part Number h5964

Automating Root-Cause Analysis:
EMC Ionix Codebook Correlation Technology vs. Rules-based Analysis
Technology Concepts and Business Considerations

Table of Contents

Executive summary	4
Introduction	4
Audience	4
Keeping pace with network complexity	4
Rules-based correlation	6
Downstream event suppression	7
Limitations of rules	8
Introducing EMC Ionix Codebook Correlation Technology (CCT).....	8
How CCT diagnoses problems	8
Dynamic generation of problem signatures	9
Behavior models and authentic problems.....	10
Example of CCT behavior models	10
Automatic creation of problem signatures	10
The robustness of CCT	11
The efficiency of CCT	11
Summary of EMC's CCT advantages.....	11
The business benefits of CCT.....	12
Conclusion	12

Executive summary

Today, more than ever, businesses depend on their Information Technology (IT) infrastructure. As a result, IT organizations must meet increasingly high standards for service assurance, a key metric that directly affects a company's profitability and business success.

At the same time, the IT infrastructure is becoming more difficult to manage. The number and heterogeneity of hardware and software elements in networked IT environments are increasing exponentially and subsequently increasing the complexity to manage a business's systems. The introduction of each new technology adds to the list of potential problems that threaten the delivery of networked-dependent services. Furthermore, as it becomes more difficult to diagnose the problems, the urgency to do so increases.

Introduction

This white paper discusses the role of root-cause analysis (RCA) in managing infrastructure availability and performance. By comparing two approaches – rules-based analysis and EMC® Ionix's Codebook Correlation Technology™ (CCT) – this paper illustrates the unique advantages of CCT and explains why CCT is the only viable method for automating the analysis of any type of problem in any networked system, no matter how complex. It also demonstrates how CCT provides uniquely powerful development- and maintenance-free intelligent analysis that automatically adapts to the managed IT infrastructure.

Audience

This white paper will be suitable for mid-to-low technical IT network administrators and operators.

Keeping pace with network complexity

As any IT manager can attest, fixing a problem is often easy, once it has been diagnosed. The difficulty lies in locating the root cause of the myriad events that appear on the management console. Studies estimate that 80 to 90 percent of downtime is spent analyzing data and events in an attempt to identify the problem that needs to be corrected.

For IT managers charged with optimizing the availability and performance of large multi-domain networked systems, it is inefficient to collect, filter, and present data to operators. Unscheduled downtime directly affects the bottom line. The need for applications that apply intelligent analysis to pinpoint root-cause failures and performance problems automatically is imperative. Only when diagnosis is automated can self-healing networked systems become a reality.

There are many types of problems that threaten service delivery: hardware failures, software failures, congestion, loss of signal, loss of redundancy, and misconfigurations, to name a few. An effective root-cause analysis technique must be capable of identifying all those problems *automatically*. This technique must work accurately for any environment and for any topology – including interrelated logical, physical, and now virtual topologies – with or without redundancy. It must be able to diagnose problems in any type of object – for example, a cable, a switch card, a server, or a database application – at any layer, no matter how large or complex the infrastructure.

Just as a doctor must diagnose a patient's disease before prescribing treatment, accurate root-cause analysis is required to determine the appropriate corrective action. If management software cannot automate root-cause analysis, that task falls to operators. Because of the size, complexity, and heterogeneity of today's networks, and the volume of data and alarms, manual analysis is extremely slow and prone to error. The intelligent analysis, adaptability, and automation of the EMC Ionix™ Codebook Correlation Technology described in this paper translate directly into major business benefits, enabling organizations to introduce new services faster, to exceed service-level goals, and to increase profitability.

The challenge of root-cause analysis

In this section, we introduce a simple example to illustrate the technical challenges of automating problem diagnosis in complex infrastructures.

Figure 1 illustrates a small-switched network with four switches (S0, S1, S2, S3) connected as a mesh for high resilience. Each switch is composed of two cards, each card contains two physical ports, and each physical port supports two logical ports. To simplify the example, we will focus on one particular problem category representative of hardware failures.

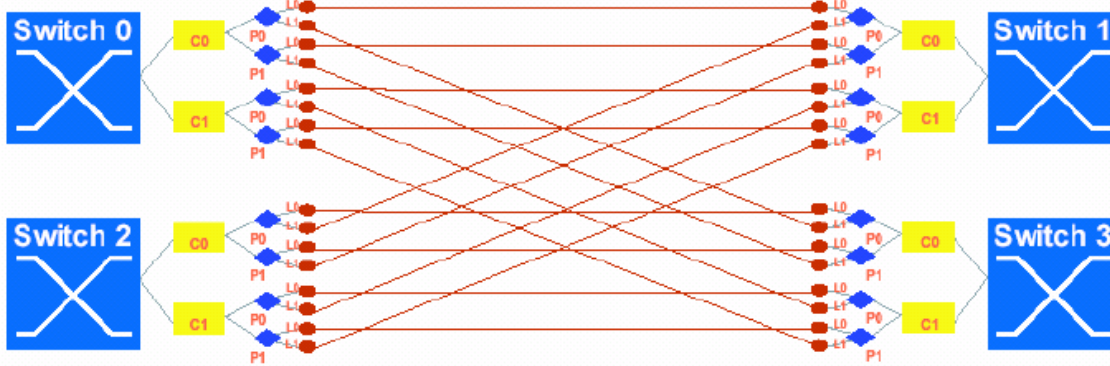


Figure 1. A simple network topology

Figure 2 illustrates a particular failure scenario: the failure of Card C0 in Switch S1, and the observable symptoms it causes. The S1C0 failure causes the symptoms listed below:

- The four logical ports layered over the physical ports on the failed card report as operationally down.
- The four logical ports in other switches that are peers of the down logical ports report as operationally down.
- The switch generates a card down alarm because of the failed card. (Note that this symptom does not always exist, as some switch vendors do not provide alarms for card failures.)

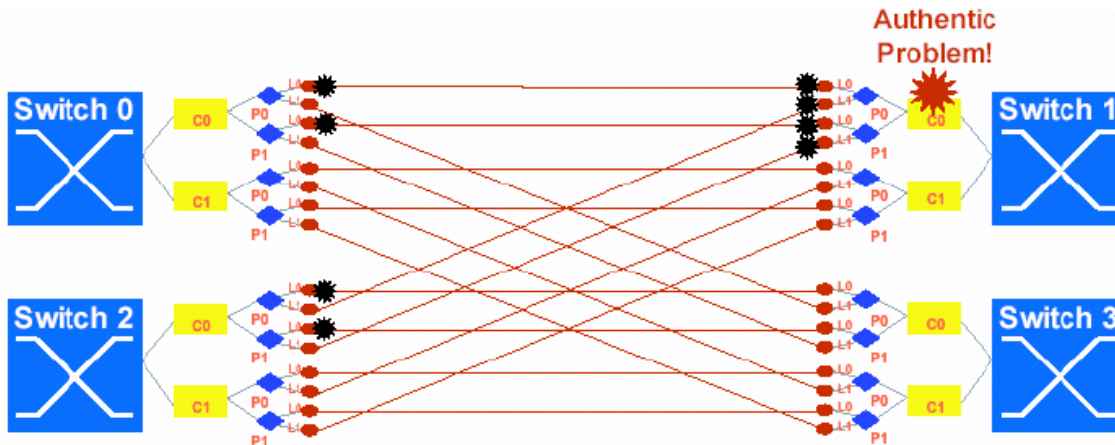


Figure 2. The signature of the S1C0 failure

This example illustrates some of the difficulties in accurately diagnosing problems:

- Problems can start in any logical, physical, or virtual (outside of this example) object in the network, attached systems, or applications.
- A single problem often causes many symptoms in many related objects.
- The absence of particular symptoms is as meaningful as the presence of particular symptoms.

-
- Different problems can cause many overlapping symptoms. For example, the operationally down status of logical port 0 over physical port 1 in Card 0 of Switch 1 could be caused by a failure in any one of the following components:
 - Switch 1
 - Switch 1 Card 0
 - Switch 1 Card 0 Port 1
 - Switch 1 Card 0 Port 1 Logical Port 1
 - Switch 2
 - Switch 2 Card 0
 - Switch 2 Card 0 Port 1
 - Switch 2 Card 0 Port 1 Logical Port 1
 - The trunk connecting Port 1 in Card 0 in Switch 1 with Port 1 in Card 0 in Switch 2.
 - Symptoms can propagate across related components. In our example, a symptom propagated from a card to its ports and from a port to a peer port. This makes it necessary to examine *all the symptoms* across related elements in order to identify the root cause.
 - Problems are not always observable in the object where they originate. For example, if the switch does not generate card failure traps, the card failure would have no direct (local) symptoms.
 - Problems can occur in objects that do not generate any observable events. For example, there is no event and/or trap associated with a trunk.

Rules-based correlation

The ability to identify the root cause *rapidly and accurately* is the core challenge of today's IT organization. Using trial and error to determine the root-cause problem wastes precious time and results in a high mean-time-to-repair (MTTR) that can have negative effects on a business and its profitability. What IT organizations require is actionable information that clearly and instantly identifies the problems that need to be fixed, so problems can be fixed before service is affected.

Event managers focused on gathering and displaying an abundance of data to users is not effective; there is far too much data already. And deduplication and filtering – the only processes that event managers can automate – offer no help in identifying whether a problem exists, and if so, what precisely is the problem. Because of the lack of intelligence in legacy event managers, users of these systems often resort to developing their own custom scripts to capture their specific rules for event processing.

The custom rules approach is a “bottom-up” development-intensive approach doomed to fail for all but the simplest scenarios in simple static networks. In this approach, the developer begins by identifying all of the events – alarms, alerts, SNMP traps, threshold violations, and so on – that can occur in the managed system and then attempts to write network-specific rules to process each of these events as they occur.

Writing rules to execute for each possible event quickly proves to be infeasible for any production network, not to mention an end-to-end networked system that includes storage, applications, servers, and networks. Even in our simple example network, almost every alarm could be a symptom of several different problems. For example, when a logical port reports operationally down, it could be the result of nine different failures of any one of nine related components. This event-at-a-time approach is clearly flawed, because the symptoms of different problems can overlap substantially, and only a particular *combination* of symptoms can distinguish one problem from another. This becomes ever more complex when virtualization is added into the environment.

A more effective rules-based approach attempting to address the root-cause analysis challenge would therefore process combinations of symptoms. For example, we could write a rule to detect the failure of Card 0 in Switch 1:

IF

"If all the logical ports that are layered over the physical ports in S1C0 report as operationally down, and all the logical ports that are their peers in connected switches also report as operationally down, then the S1C0 failure is the root-cause problem generating all these symptoms."

S1C0P0L0 down AND
 S1C0P0L1 down AND
 S1C0P1L0 down AND
 S1C0P1L1 down AND
 S0C0P0L0 down AND
 S0C0P1L0 down AND
 S2C0P0L1 down AND
 S2C0P1L1 down

THEN CONCLUDE S1C0 failure

Figure 3. One of the rules for detecting the S1C0 failure

Because there are at least 60 possible failures (failures of every switch, card, physical port, and logical port) in this network, analyzing simple failures in this tiny four-switch network requires at least 60 complex rules like the one above. And this is just the beginning! What happens if more than one failure occurs at the same time? What happens if a network problem causes a delay or data loss, and, as a result, a symptom is missed? What happens if a resilient architecture masks failures by dynamically reconfiguring around them, so that there are no alarms or disruptions in service? Failures that reduce the degree of resilience require immediate diagnosis and repair, because a subsequent failure may bring the service down. The number of rules needed to address all these situations grows exponentially with the size of the network.

Clearly, an organization willing to invest the effort necessary to write rules faces enormous challenges. Typically, the number of network devices is in the hundreds or even thousands, and each device is far more complex than the switches in our simple example. The number of rules required for a typical network, without accounting for delay or loss of alarms, or for resilience, can easily reach millions. The development effort necessary to write these rules would require many person-years, even for a small network.

To make matters worse, changes in the network configuration can render some rules obsolete and require the writing of new ones. For example, in our four-switch network, if the logical ports of Card 0 on Switch 1 were peered with the logical ports of Card 1 instead of Card 0 on Switch 0, then rules would have to be modified. Moreover, the rule-based approach does not really solve the problem, since when symptoms are lost or delayed, rules will fail to pinpoint the failure. Thus, at the point in time when their proper functioning is needed most, that is, when network problems are causing loss and delay, rules-based systems are least reliable.

As a result of these deficiencies and the overall complexity of development, attempts to add intelligent rules to an unintelligent event manager have not been successful in practice. In fact, rules-based systems have consistently failed to deliver a return on investment (ROI) associated with the huge development effort.

Downstream event suppression

What some management vendors call root-cause analysis is actually a form of *downstream event suppression*. Downstream suppression is a path-based technique for reducing the number of alarms to process when analyzing hierarchical IP networks. Downstream suppression works as follows: a poller periodically polls IP devices to verify that they are reachable. When a device fails to respond, downstream suppression does the following:

- Ignores failures from devices downstream (farther away from the poller) from the first device.
- Selects the device closest to the poller that fails to respond as the “root cause.”

There are several flaws in this technique. Downstream suppression requires that the network have a simple hierarchical architecture, with only one possible path connecting the poller to each device. However, today’s mission-critical networks all leverage redundancy to increase resilience. Downstream suppression

Automating Root-Cause Analysis:

does not work in redundant architectures because the relationship of one node being downstream from another is undefined: there are multiple paths between the manager and managed devices. Even our simple four-switch network has no notion of downstream.

In a resilient network, all devices remain reachable following a single failure. Thus, there are no ping failures to suppress and no symptoms to correlate. In fact, operations staff members are not even aware there was a failure and are not alerted to the increased risk.

In conclusion, the application of downstream suppression to today's networks is extremely limited. This technique applies only to simple hierarchical networks with no redundancy, and addresses only one problem, IP node failure. Because of these limitations, downstream suppression offers little in the way of automating problem analysis, and certainly cannot claim to offer root-cause analysis.

Limitations of rules

The following list summarizes the limitations of rules-based correlation.

High development and maintenance costs

A rules-based solution requires a major development effort to write the rules. The number of rules required grows exponentially with the size of the network, making it all but impractical for non-trivial networks. Rules must be developed and maintained consistent with network changes as well as with any new type of elements and/or events. The dependence of rules on a changing environment makes rules-based systems a never-ending maintenance nightmare.

Inadequate performance and scalability

Rules execution is inherently slow. When an event is detected, all rules pertaining to that event must be looked up and executed within the realtime correlation path. Moreover, rules execution does not always converge to a conclusion.

Poor accuracy and reliability

Because rules-based correlation requires an exact match of rules to events, it is unreliable when a network is overloaded or experiencing problems. But these are precisely the times when it is critical that the correlation function properly.

In summary, the rules-based approach takes IT organizations a step beyond event consolidation and filtering, *but it cannot address the challenge of automating root-cause analysis of authentic problems in mission-critical networks*. Organizations whose business depends on networked systems require a better approach to event correlation.

Introducing EMC Ionix Codebook Correlation Technology (CCT)

CCT is a mathematically founded next-generation approach to automating the correlation required for service assurance. CCT is unique in its ability to:

- *Automatically* analyze *any* type of problem in *any* type of physical, logical, or virtual object in *any* complex environment
- Build intelligent analysis into off-the-shelf solutions
- *Automatically* adapt the intelligent analysis to the managed environment even as it changes
- Provide instant results

How CCT diagnoses problems

The premise of CCT is simple: Each problem in a networked system has a unique *signature* — the symptoms that it causes. This signature is the key to identifying the problem.

A signature typically contains many symptoms - symptoms in the faulty component where the problem occurs and symptoms in related components that are affected by the original problem. Because symptoms of different problems overlap, it is the *unique combination* of symptoms that differentiate one problem from another.

CCT diagnoses problems in real time by matching symptoms to problem signatures. The problem whose signature most closely matches the incoming data is identified as the problem. Figure 2 illustrates the signature when Card C0 in Switch S1 fails. When CCT sees this particular combination of symptoms, an exact match to the signature of Card C0 in Switch S1 failing, it automatically concludes that Card 0 in Switch 1 has failed.

The principle of diagnosis in CCT is similar to a medical diagnosis in which a doctor matches the combination of symptoms exhibited by a patient to symptoms of known diseases. The doctor identifies the disease based on symptoms. In this process, the doctor accounts for the symptoms a patient does not exhibit as well as those symptoms that he/she does exhibit.

Because CCT's root-cause correlation consists of a simple comparison of symptoms to signatures, it is extremely fast. Moreover, because CCT looks for the closest match and not necessarily an exact match, it can diagnose accurately even with incomplete information, if, for example, some of the symptom data is delayed or lost.

Dynamic generation of problem signatures

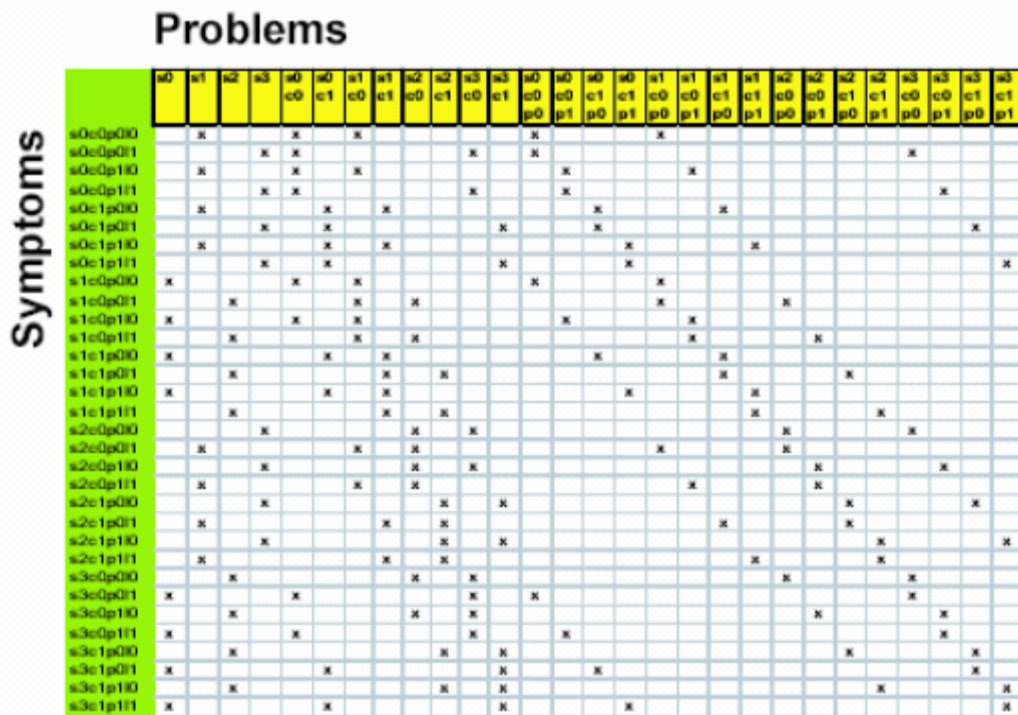


Figure 4. Problem signatures for the network in Figure 1

Signatures are unique to each infrastructure, and each infrastructure is unique and changes over time. To manually create the signatures for a network would be as tedious as developing rules. But CCT introduces breakthrough technology to solve a previously unsolvable problem: CCT introduces an algorithm to

automatically compute signatures for each problem occurrence and keep them current as the topology changes.

Behavior models and authentic problems

CCT's automatic computation of signatures leverages generic object-oriented *behavior models*. The object-oriented behavior models describe classes of objects and their associated problem behaviors. The key to CCT's description of object classes and their behaviors is that the description is independent of the infrastructure topology.

In developing behavior models, CCT applies a "top-down" methodology focused on authentic problems. These are the critical problems whose occurrence threatens service delivery. In order not to impact the service, authentic problems should be sought proactively and resolved as soon as possible.

The methodology for building CCT behavior models is to start by identifying classes of objects to manage — logical, physical, and virtual. Then, for each such class, identify the authentic problems associated with that class, and the corresponding symptoms of each such problem. Symptoms include any observable event, such as alarms, traps, expressions over MIB variables, other instrumented values, or any other external signal.

CCT behavior models describe two types of symptoms. Symptoms directly associated with the faulty object are referred to as *local symptoms*. For example, "server unreachable" is a local symptom observed in the server that failed. Symptoms that appear in objects related to the faulty object are referred to as *propagated symptoms*. For example, "application unavailable" is a propagated symptom that appears in applications that run on a server that has failed. Symptoms that originate in one object but appear in a related object or objects make problem diagnosis especially difficult.

Example of CCT behavior models

CCT behavior models specify the authentic problems associated with each class, each problem's local symptoms, and the symptoms that propagate to directly related objects. The following example illustrates the CCT behavior models used to analyze a switched network like the one in Figure 1. We describe behaviors of three classes of objects: cards, physical ports, and logical ports. Because our application example is focused on managing availability, the set of authentic problems for each of type of object contains a single problem called "down."

Card Behavior Model

Problem *Down Causes PortDown*

Propagated symptom *PortDown To Physical Ports in the Card Down*

Physical Port Behavior Model

Problem *Down Causes PortDown*

Propagated symptom *PortDown To Logical Ports Layered over the Port are Down*

Logical Port Behavior Model

Problem *Down Causes OperationallyDown, ConnectedPortDown*

Local symptom *OperationallyDown*

Propagated symptom *ConnectedPortDown To Connected Logical Ports Down*

Automatic creation of problem signatures

To create signatures automatically, CCT combines the *generic* information in object behavior models with *specific* information about the managed infrastructure's topology. CCT automatically computes signatures for all authentic problems that can occur in the infrastructure using the following methodology:

1. Creates an instance of each object in the infrastructure based on the class for that object. For the network in Figure 1, a switch object would be instantiated for each switch, a card object for each card, and so on. Also, the relationships between instantiated objects, for example, between a physical port

Automating Root-Cause Analysis:

EMC Ionix Codebook Correlation Technology vs. Rules-based Analysis
Technology Concepts and Business Considerations

and the logical ports layered over it, would also be instantiated. The result is a semantics-rich repository of information that represents the current infrastructure, including all logical and physical objects and their interrelationships.

2. Computes a signature for each occurrence of each potential authentic problem in the infrastructure by traversing the repository starting from the object where that problem originates and collecting all the problem's symptoms. This includes local symptoms directly associated with that object and the propagated symptoms. The resulting signature is then stored in a Codebook.

A similar process is used to automatically update problem signatures whenever the topology changes.

Figure 4 depicts signatures generated for the network in Figure 1 from the behavior models above. Each column represents a distinct problem, and each row represents a symptom. An "x" appears at the intersection of a problem and a symptom if that problem causes that symptom. The computed signatures are used to correlate data and events streaming into the CCT correlation engine.

The robustness of CCT

Unlike a rules-based system, CCT does not require an exact match to diagnose a root cause. When presented a subset of symptoms, CCT produces the "best explanation" for the observed symptoms. When presented with an incomplete set of symptoms, CCT assigns a probability to one or more root causes whose signatures are closest to the observed events. This probability is computed using the mathematical distance between the observed symptoms and the signature. Because of its correlation process, CCT can correlate symptoms and identify root causes quickly and accurately, even if symptoms are delayed and lost, or if unexpected symptoms occur. These situations can occur when a network experiences problems.

The efficiency of CCT

CCT is the most efficient correlation technique available. Its complexity is, at most, linear in the number of managed objects. This is because the number of signatures to match is proportional to the number of problem instances. Moreover, because correlation is simply a vector-distance computation, it is significantly faster than searching, retrieving and executing the rules that pertain to an event each time it occurs.

CCT has additional features that make it extremely fast while optimizing its use of computing and communications resources.

1. The focus on *authentic problems* means that CCT monitors only those symptoms that indicate *authentic problems*, not every event and alarm. Thus, CCT's efficiency is much higher than that of any other event manager, including those that perform only filtering and de-duplication.
2. CCT only requires a subset of the available symptoms to perform correlation. Because CCT only requires that each problem instance have a unique signature, it is often sufficient to monitor a subset of available symptoms for accurate diagnosis. The process of selecting the optimal set of symptoms to monitor is called Codebook Reduction.

Summary of EMC's CCT advantages

CCT provides the only automated, accurate realtime analysis of root-cause problems and their effects in networked systems. Its many advantages include:

- No development
CCT supports off-the-shelf solutions that embed intelligent analysis and automatically adapt to the environment. Today, there is a rich set of solutions available from EMC Ionix IT Operations Intelligence (formerly EMC Smarts[®]) that address the availability and performance of critical environments. These solutions provide tremendous benefits out of the box.

Users who want to augment these solutions can do so with great efficiency. The development effort consists of expanding existing models with new classes and symptoms independent of the size of the managed environment.

- **No maintenance**
Because the analysis logic is automatically generated, CCT solutions dynamically adapt to topology changes. This eliminates the high maintenance costs required by rules-based systems that demand continual reprogramming.
- **Performance and scalability**
Because CCT consists of a simple distance computation between observed events and problem signatures, CCT solutions perform astonishingly fast, correlating many hundreds of events per second. In addition, CCT is highly efficient in its use of computing resources and network bandwidth, because it monitors only the symptoms that are needed to diagnose *authentic problems*.
- **Accuracy and robustness**
Because CCT looks for the closest match between observed events and problem signatures, it can reach the correct conclusion even with incomplete information – for example, when events are delayed or lost.
- **Out-of-the-box applications**
Because the correlation logic is automatically computed from reusable object-oriented behavior models and network-specific topology, Codebook applications work out-of-the-box, adapting to the networked IT environment without any development or maintenance.

The business benefits of CCT

Leveraging CCT to automate service assurance provides substantial business benefits, including:

- **Faster time-to-new-service**
Because CCT automatically generates its correlation logic for each specific topology, new services can be managed immediately and new customers can be added to existing services without delay.
- **Greater availability and performance of business-critical systems**
Automated service assurance reduces downtime and performance problems that can affect revenues, productivity, customer satisfaction, and overall financial performance. A better performing infrastructure makes a positive contribution to the bottom line.
- **Improved productivity of scarce resources like time, talent, and management attention**
By eliminating the need for development, ongoing maintenance, and manual diagnostic techniques, CCT enables IT organizations to be proactive and to focus their attention on strategic initiatives that increase revenues and market share.
- **Freedom to adopt new technology**
CCT provides a future-proof foundation for managing any type of complex infrastructure. This gives CCT users the freedom to adopt new technology, with the assurance that it can be managed effectively, intelligently, and automatically.

Conclusion

This white paper has examined the role of automated realtime root-cause analysis in managing infrastructure availability and performance and compared two approaches, rules-based analysis and Codebook Correlation Technology.

Through an in-depth explanation of the unique advantages of CCT – intelligent analysis, adaptability, and automation – the paper has demonstrated that CCT is the *only* viable method for analyzing any type of problem in any technology. CCT’s many features translate directly into major business benefits that enable organizations to introduce new services faster, exceed their service-level goals, increase their profitability, and stay ahead of the technology curve without risk.