

Parallel Sysplex: Achieving the promise of high availability

Session 17431 13 August 2015

Mark A Brooks

mabrook@us.ibm.com

z/OS Sysplex Development

IBM Poughkeepsie, NY



SHARE is an independent volunteer-run information technology association that provides **education, professional networking and industry influence.**

Copyright (c) 2015 by SHARE Inc.  Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>





Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM®	MQSeries®	S/390®	z9®	IBM® (logo)
ibm.com®	MVS™	Service Request Manager®	z10™	AIX® BladeCenter®
CICS®	OS/390®	Sysplex Timer®	z/Architecture®	DataPower®
CICSPIlex®	Parallel Sysplex®	System z®	zEnterprise™	DS4000®
DB2®	Processor Resource/Systems Manager™	System z9®	z/OS®	DS6000™
eServer™	PR/SM™	System z10®	z/VM®	DS8000®
ESCON®	RACF®	System/390®	z/VSE®	POWER7®
FICON®	Redbooks®	Tivoli®	zSeries®	ProtectTIER®
GDPS®	Resource Measurement Facility™	VTAM®	zEC12™	Rational®
HyperSwap®	RETAIN®	WebSphere®	Flash Express®	System Storage®
IMS™				System x®
IMS/ESA®	Geographically Dispersed Parallel Sysplex™			XIV®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license there from.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

InfiniBand is a trademark and service mark of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Abstract

A properly configured parallel sysplex can deliver near continuous availability. "Properly configured" is often mistakenly equated with redundancy. But installations with highly redundant configurations have suffered significant outages. Why wasn't redundancy enough? What went wrong? In part 1, the speaker shares concepts and principles gleaned from analyzing customer installations under the auspices of IBM's High Availability Center of Competency (the HACoC team identifies risks to HA and suggests solutions). Part 2 provides practical application of these principles with suggestions for configuring and operating your sysplex in a way that maximizes the likelihood of achieving your availability goals for your critical business services.

IBM High Availability Center of Competency (HACoC)

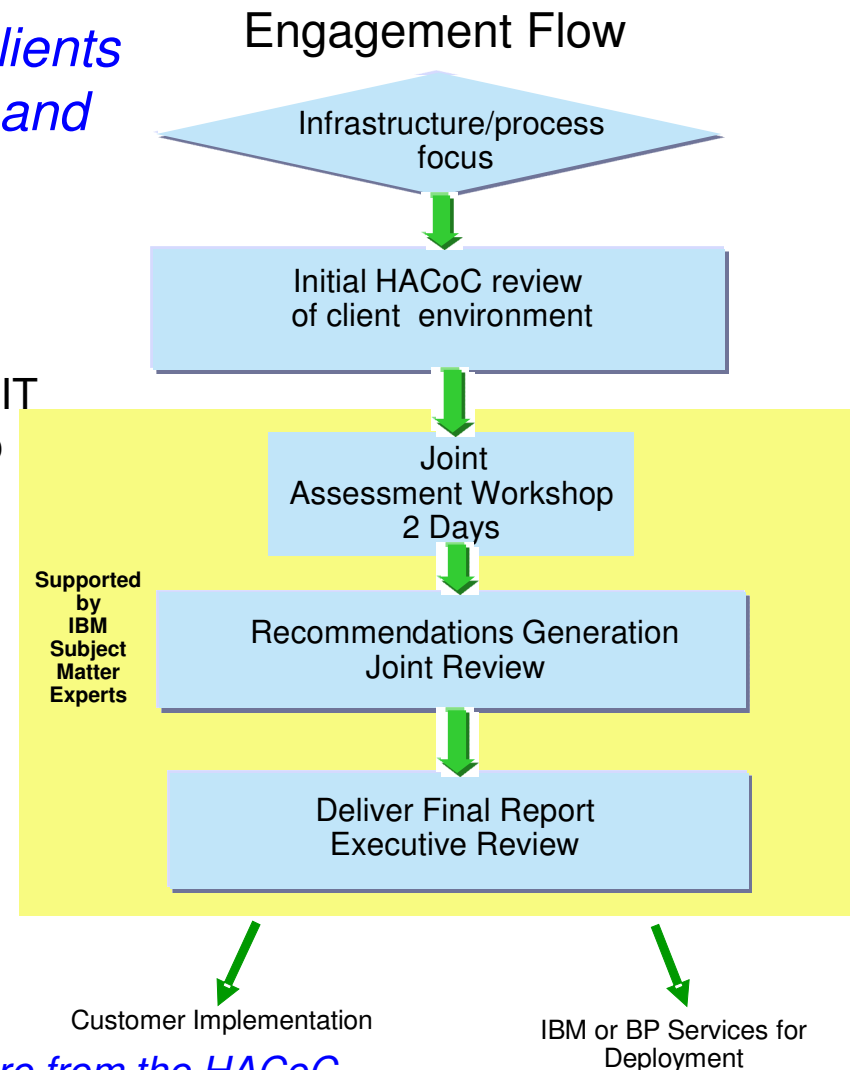
The mission of the HACoC is to help clients achieve their service availability goals and maximize the value they receive from IBM systems and storage products.

- Along with IBM subject matter experts, the HACoC engages with clients for a high-level review of their IT architecture and service management processes to identify issues that may affect service availability.

These may include:

- *Single Points of Failure*
 - *Product gaps*
 - *Inhibitors*
 - *Test / solution opportunities*
 - *High priority focus areas*
- The HACoC provides tactical and strategic recommendations to help clients achieve their availability goals.

Many of the concepts described in this presentation are from the HACoC. For more information about the HACoC and how they can help you improve availability, send an email to hacoc@us.ibm.com.



Trouble in paradise

- Companies face increasing pressure to deliver business services 24x7
- A properly configured parallel sysplex can deliver near continuous availability
- And yet, installations with highly redundant sysplex configurations fail to meet their availability objectives
 - Sometimes suffering significant detrimental business impact
- What's going wrong?

Terminology

- **High Availability (HA) masks unplanned outages from end users**
 - The attribute of a system to provide service during defined periods, at acceptable or agreed upon levels. HA is achieved through:
 - Fault tolerance
 - Automated failure detection, recovery, bypass, and reconfiguration
 - Thorough testing and effective problem and change management

- **Continuous Operations (CO) masks planned outages from end users**
 - The attribute of a system to continuously operate. CO is achieved through:
 - Non-disruptive changes to hardware, software, and configuration
 - Software coexistence

- **Continuous Availability (CA) masks both planned and unplanned outages from end users**
 - The attribute of a system to deliver non-disruptive service to the end user 24 hours a day, 7 days a week.

Terminology

- **IT Resilience** - Attribute of IT systems and applications to ensure high service availability through monitoring and automatic adjustment of redundant or virtualized infrastructure components.
 - Arises from the implementation and management of an IT resiliency plan.
- **Business Resilience (BR)** – The ability of the business to rapidly adapt and respond to opportunities, regulations, and risks in order to: maintain secure, continuous business operations; be a trusted partner; enable growth
 - Business Resilience spans business strategy, organizational structure, business and IT processes, core business and IT infrastructure, applications and data, and facilities.
 - Business Resilience demands IT Resilience.

High Availability is not some number nor is it about component availability; it's a concept focused on the availability of the service being delivered to the business.

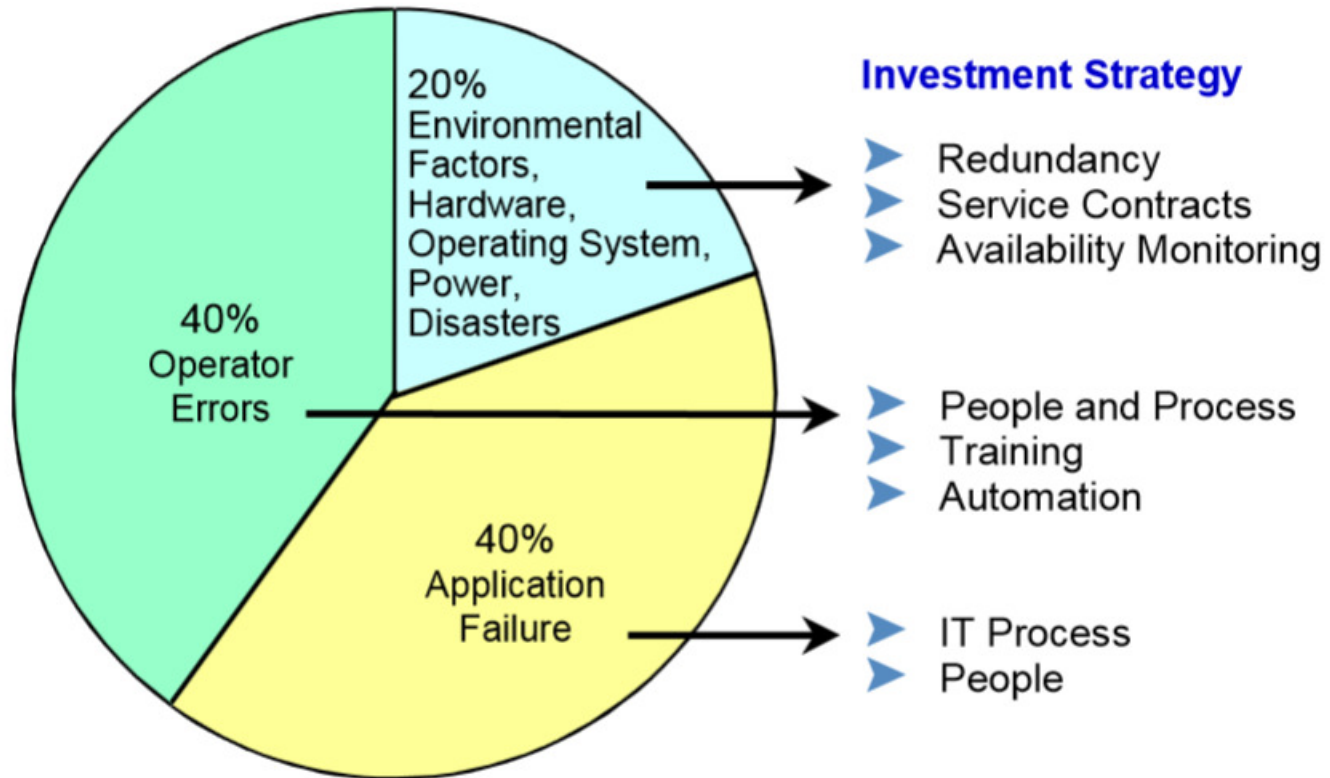
High Availability and Disaster Recovery address different aspects of the Business Resilience landscape

	(High) Availability	Disaster Recovery
Triggering Event	Component failure within a datacenter	Event that results in a full Datacenter outage
Objective	Minimize or mitigate the duration and scope of impact. Restore full service within the datacenter.	Re-establish critical services at an alternate site.
Typical Recovery Objectives	<ul style="list-style-type: none"> • SRO = seconds or minutes • RPO = 0 <p>SRO “clock” starts when incident occurs</p>	<ul style="list-style-type: none"> • RTO = hours • RPO > 0, measured in minutes <p>RTO “clock” starts when disaster is declared</p>
<p>SRO = Service Restoration Objective RTO = Recovery Time Objective RPO = Recovery Point Objective</p>		

Our goal is to improve availability of business services

- The business needs to avoid service outages
- A typical question:
 - “How can you guarantee there won't be another failure?”
 - Answer: You can't. Failures will occur.
- A more helpful question:
 - “How can we mask failures so that critical business services appear to be highly available?”
 - Recognizes that failures will occur
 - Maintains focus on the end user
 - And naturally leads to secondary questions with quantifiable answers that can help guide business decisions
 - Critical services? Availability requirement? Cost if not met?
 - IT dependencies? Potential failures? Ways to mitigate? Costs?
 - Right architecture? Better way?

Causes of Unplanned Outages



Causes of application downtime and appropriate responses

Source: Enterprise Guide to Gartner's High-Availability System Model for SAP, R-13-8504 dated 12/20/01
<http://www.tarrani.net/mike/docs/HiAvailModel4SAP.pdf>

Challenges to achieving high availability of business services

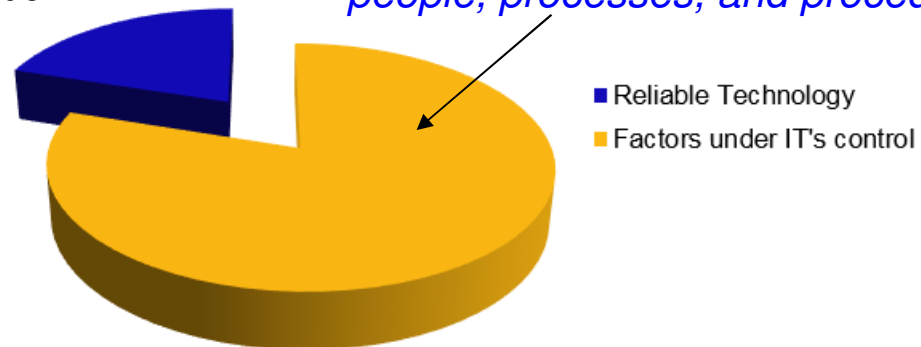
- Complex mix of IT standards
- Single points of failure
- Unsupported technologies
- Limited disaster recovery
- Gaps in End to end Monitoring: difficulty determining source or extent of problems
- Lack of automation: alerts, responses
- Failure to respond to alerts
- Affinities
- Change impacts
- Incident handling: outages may be elongated
- Managing capacity and growth
- Lack of proactive maintenance
- Conflicting management tools: complex business systems can lead to complex problems
- Lack of integration
- Lack of flexible, dynamic, or responsive components
- Unaligned business and IT service objectives
- Resources: lacking or stretched
- Cost of downtime versus availability needs

Availability of IT services requires more than installing reliable technology. You must also consider:

- How the Technology is implemented
- How the applications are designed, deployed, and integrated with the underlying resilient infrastructure
- How the service is managed

HACoC Experience:

75-80% of unplanned service outages are the result of issues related to people, processes, and procedures



An Architecture for Achieving Highly Available Business Services

- **Reliable** - **minimize chance of failure**
- Resilient - minimize chance of service outage
- Recoverable - minimize duration of service outage
- Resourceful - minimize the damage

Achieving HA through reliability

- An obvious way to avoid service outages is to eliminate the problems that induce them
 - Many IT shops take this approach
 - Even perform “root cause” analysis of triggering problem
 - Sadly, some analyze but do not act
- It is both improbable and unrealistic to expect that all problems can be eliminated
 - more on that later
- But installations can improve the likelihood of achieving their availability goals by trying to minimize the chance of failure ...

Reliable: Minimize chance of failure

- **Comprehensive testing**
 - Remove defects prior to production
- **Regular maintenance**
 - Policy with rationale
 - Periodic review of effectiveness
- **Effective change management**
 - Facilitates successful changes
- **Effective problem management**
 - Iteratively improve system, processes, and skills

Testing

- System volume and stress;
- Production-like platform and data;
- Test scripts and Transaction Driver;
- Failure injection and recovery testing

Release Management

- Establish currency policies
- Deploy into production
- Review functional upgrades
- Package into testable releases

Change Management

- Enable growing quantity of changes;
- Measure Effectiveness by function and educate;
- Risk assessment and mitigation;
- Exception handling for emergencies and business needs;
- Accountability lies with developer;
- Readiness Checklists and implementation scripts;
- Focus on Quality of Change itself

Problem Management

- Quick analysis (Cause of incident) and timely correction;
- Track corrections and escalate exceptions;
- Reduce recurring incidents;
- Knowledge DB;
- DB structure supports reporting needs
- Problem Prevention - True root cause (Cause of defect);
- Pursuit of secondary contributors;
- Failure Pattern and Trend Analysis

Hardware eventually fails. Software eventually works.

Elimination of all software defects is unrealistic

- z/OS has at least $2^{1.2 \text{ million}}$ unique paths through its code
 - FYI: Roughly 2^{89} nanoseconds have elapsed since the big bang
 - To discover a defect, must execute the code path that expresses it
 - Would need to execute all such paths to prove zero defects
 - Majority of APARs are reported once and never again
 - Factors affecting error discovery
 - Rarity – not all paths execute with same probability
 - Complexity – number of conditions needed to trigger the path
- IBM is working to better simulate customer workloads in our test environments
 - Increases the likelihood that we execute customer code paths
- Reality: you are the only one running your workload
 - There will be failures that are unique to you
 - Odds of uniqueness increases as defect rates decline

Elimination of all hardware problems is improbable

- Probability that a system with MTBF of N years will NOT fail by end of year T is: $R(T) = e^{(-T/N)}$

MTBF*	Year 1	Year 2	Year 3	Year 4	Year 5
50 yrs	98.02%	96.08%	94.17%	92.31%	90.48%
100 yrs	99.00%	98.02%	97.04%	96.08%	95.12%
1000 yrs	99.90%	99.80%	99.70%	99.60%	99.50%
10,000 yrs	99.99%	99.98%	99.97%	99.96%	99.95%

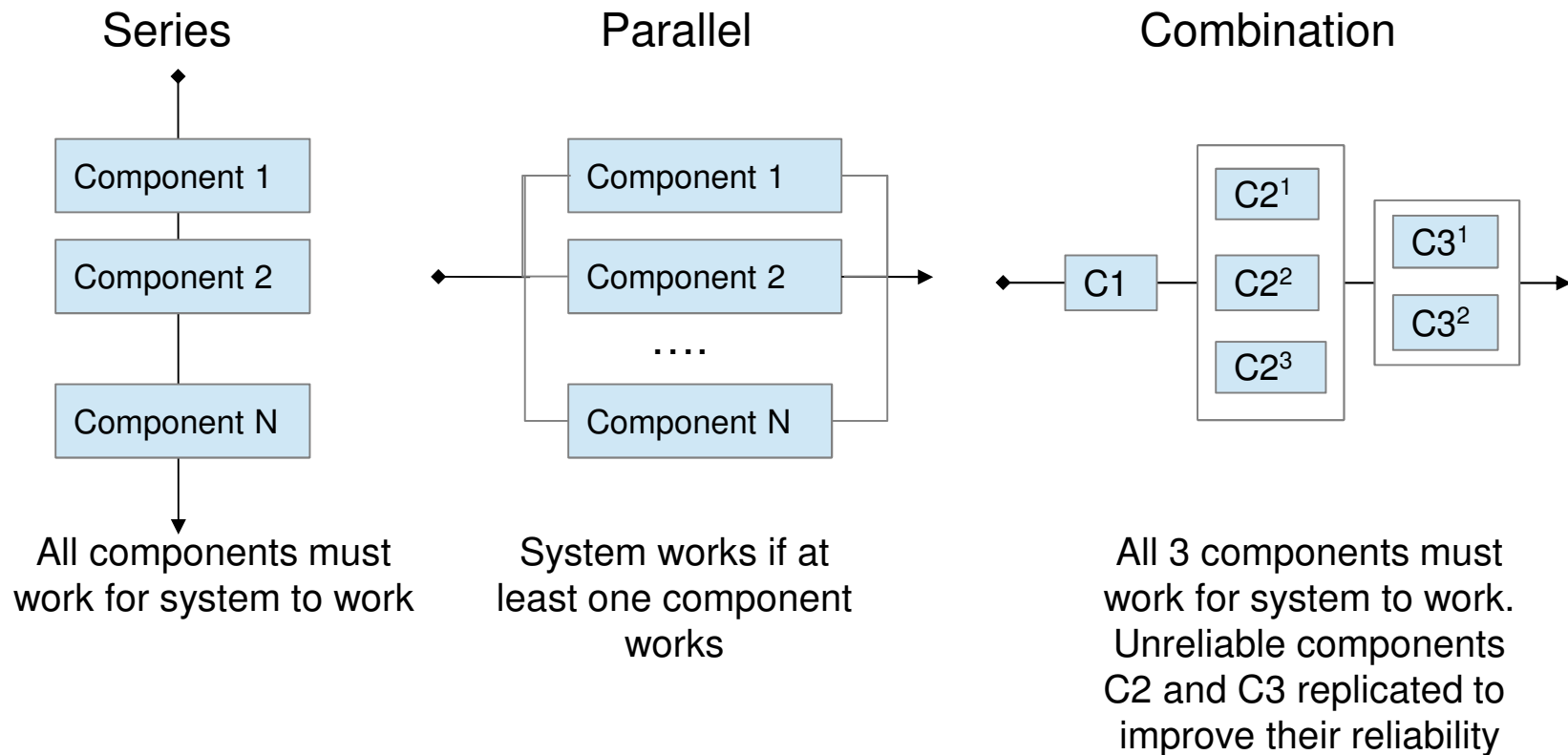
As reliable as the hardware may be, the chances of failure are not zero.

Mean Time Before Failure (MTBF) is a statistical value. It is neither a prediction nor guarantee of how long any given system will last before failing. We cannot know how long a specific system will last before it fails. But we can use MTBF to determine the probability that it will survive a given time period before it fails.

** I found various sources on the web stating mainframe MTBF is 20-50 years. I do not know if these are accurate. My point is made even if it's longer. To my knowledge, IBM does not publish MTBF data.*

Reliability of a system of components

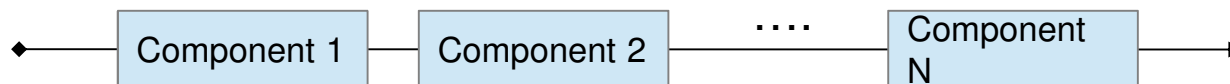
- The reliability of a system consisting of multiple components depends on the reliability of each component and how the components are assembled



Reliability of system with series of components

- Reliability of system is product of reliability of each component
 - System of two components that are 97% reliable is 94% reliable.
- The system becomes less reliable as number of components increase
 - A system of 50 components that are each 99.9% reliable is only 95% reliable; a 100 component system would be 90% reliable
- To improve series system reliability:
 - Use more reliable components
 - Reduce the number of components

How many components must function for your business services to function?



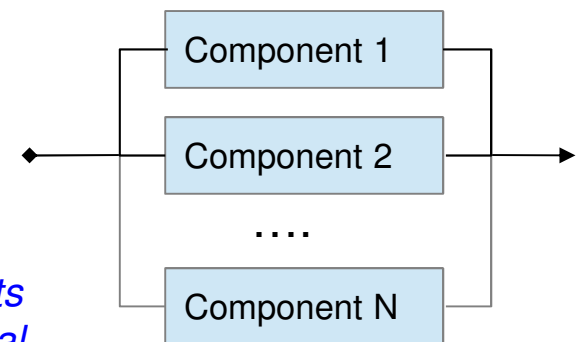
System fails if any one component fails

Reliability of system with parallel components

- Reliability of system is determined by the probability that all of the individual components fail at the same time:
 - $1 - \prod_{i=1..N} (1 - r(i))$ where $r(i)$ is reliability of component i
 - So a parallel system of 2 components that are each 97% reliable, will have reliability of 99.91% $\{1 - (.03)^2 = .9991\}$
 - With 3 components, 99.99%; with 4, 99.9999%; with 5, 99.999998%
- Can reach a point of diminishing returns
- To improve parallel system reliability
 - Use more reliable components
 - Increase number of components

Probability that system works is 1 minus the probability that all of the components fail simultaneously

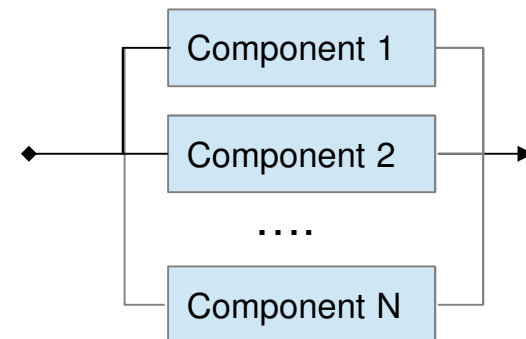
Probability that all components fail is product of their individual failure probabilities.



System fails if all components fail

Reliability of system with parallel components ... Real world considerations

- Each component must be equally capable of performing the required service
 - Otherwise component “i” does not provide redundancy for “j”
- Will failure detection and recovery complete before business services are impacted?
 - Failure detection may not be immediate. Timeouts perhaps?
 - May need to isolate failed component before survivors can continue
 - Failure during in-flight transaction may need recovery actions
- Capacity concerns
 - Might need at least $K > 1$ of N components to function in order to meet capacity or performance requirements of the business service



System fails if all components fail

An Architecture for Achieving Highly Available Business Services

- Reliable - minimize chance of failure
- **Resilient** - **minimize chance of service outage**
- Recoverable - minimize duration of service outage
- Resourceful - minimize the damage

Despite all our efforts to eliminate them, failures will occur. When that happens, we must try to mask the failure so that business services are not disrupted ...

re·sil·ient – adjective

- 1. springing back; rebounding.
- 2. returning to the original form or position after being bent, compressed, or stretched.
- 3. recovering readily from illness, depression, adversity, or the like; buoyant.

- **Resilient does not equal error free.** Single component failures will occur. Given this fact, our goal is to prevent a single component failure from impacting availability of business services to end users
- A resilient IT infrastructure is configured to achieve desired availability, scales to meet the needs of an enterprise, adheres to best practice operational procedures, and leverages all available technology to recover from issues quickly.

Avoiding service disruption through fault tolerance

- Requires robust architecture and implementation ...
 - Redundancy and clustering to avoid single points of failure
 - Sufficient capacity (white space) to take on work of failed component
 - Extensive use of automation to:
 - Monitor events
 - Initiate corrective actions
 - Automatic recovery that is sufficiently transparent
 - Component bypass, reconfiguration, failover, restart, ...
- With fault tolerance built into every component of every layer that supports the business services
 - The integrated system must be fault tolerant from end to end
 - Implies the applications must be fault tolerant as well !
- And ...

SPOF analysis often inadequate !
Affinities, data, ...

Avoiding service disruption through fault tolerance ...

■ Skilled staff obsessed with HA ...

- Culture of proactive prevention
- Knowledgeable
 - Requirements
 - Business impacts
 - Failure points and scenarios

Being proactive, doing true root cause analysis, and “closed loops” are critical.

■ Who have excellent service management processes

- Availability management
- Problem management
- Event management
- Recovery development
- Test
- Service level management
- Configuration management
- Capacity planning
- Operations management

Component failure does not disrupt business services

Ideally, a fault tolerant architecture and infrastructure allows service to continue uninterrupted despite component failure.



- Architect a technology solution
- Design in fault tolerance at every layer
- Build with components that support HA
 - Systems
 - Storage
 - Applications
 - Network
- Ensure redundant capacity
- Provide capability to remove any component at any time without service impact.
- Provide automation to quickly restore services.

Design, Build, Maintain
the Technology for HA

High Availability requires excellence across all areas that support the business services: Technology, Processes, People.

- Assign Organizational functions
 - Proactive culture
 - Service Planning (linking development with delivery)
 - Availability Management
 - Enterprise Architecture (Systems and Applications)
 - The right skills at all support levels
- Define Effective Processes
 - Architecture and standards
 - Developing HA solutions
 - Testing and validation
 - Managing the technology
 - Implementing updates
 - Fast service restoration
 - Correcting errors
 - Proactive prevention
- Provide information for sound management decisions
 - CMDB and component status
 - Services catalog and business impact
 - Change history
 - Cost of down time and down time history
- Prepare ability to remove any component at any time without service impact.

Design, Build, Manage
the Service for HA

Parallel Sysplex

- Sysplex is about eliminating “single points of failure” by providing redundancy for all sysplex components:
 - Servers/CECs
 - z/OS Systems
 - DASD Controllers
 - Coupling Facilities
 - Links
 - Middleware regions
 - Application regions
 - etc.
- In order to effectively mask failures in the sysplex:
 - Redundant components must be “clones” so that any survivor is capable of processing the same work as the failed instance
 - Work must be able to flow freely to the surviving instances
 - The resources needed to process work must be accessible from wherever the surviving instances happen to run
- With this infrastructure, we can minimize the chance that business services will be impacted by a component failure
 - However, potential for service disruption will never be zero

Parallel Sysplex ...

- In other words, work must be capable of running anywhere in the sysplex
 - On any active system
 - On any active middleware instance
 - On any active application region
- If the sysplex has redundant components which are not “clones” of one another ...
 - Whether due to affinities, lack of workload routing capabilities, software licensing restrictions, capped processors, etc.
- Then that redundant infrastructure ***will be ineffective*** in providing high availability
 - Service disruptions will occur

Lack of fundamental architecture and process activities inhibits HA

**Many factors typically contribute to service disruptions when HA objectives not met.
Technology failures are only a small part of the problem.**

Unnecessary unplanned service outages.

- Lack of HA Architecture
- Lack of maintenance
- Changes cause disruption.
- Problems not adequately fixed the first time
- Problems not proactively prevented
- Human errors

Service outages longer than necessary.

- Lack of HA architecture
- Lack of planning and preparation.
- Lack of skills and documentation.
- Lack of failure recovery testing

Applications contribute to service outages.

- Lack of HA architecture
- Failures – poor quality
- Designs that limit or inhibit mobility
- Limited exploitation of features
- Ineffective integration testing

Unprotected data

- Lack of HA architecture
- Unprepared for data corruption

Availability of IT services requires much more than simply installing reliable technology.

Design, Build, and Maintain the Technology for HA

- Redundant Hardware components
 - Processors, CFs, Links, IO, Storage
- Redundant Software components
 - Including middleware and application components
- No single points of failure
 - Find and eliminate workload SPOFs/affinities
- Dynamic routing capabilities for all workload
 - No static routing affinities
- Data sharing for all critical data
 - No data-related affinities
- Automated Recovery / Restart
 - Aggressive sysplex automation and alerting

Production-like Volume Testing

Separate from production sysplexes

Good operational processes and procedures, especially for sysplex problem determination and recovery

Take advantage of latest tools and technology

Simplification/cloning of the environment – symmetry and “anything runs anywhere”

Sufficient failover capacity for recovery

“White space” and/or automated CoD

Processors, memory, I/O

z/OS capacity and CF/link capacity

Maintenance strategy

Stay up to date, avoid defect rediscoveries

Health Checks with follow-up

The z/OS stack is capable of HA. Is it configured to run that way?
Are your applications capable of HA (sysplex enabled)?

Design, Build, and Manage Service for HA

- **Proactive**
 - Understand and plan
 - Justify and build
 - Analyze and fix
- **Application architecture**
 - Fault tolerant
 - Sysplex enabled
- **Effective processes**
 - Closed loops
 - Spawn improvements
- **Failure injection/testing**
 - Practice, verify, improve

Service Level Management

- Business Requirements;
- Common IT Service Level Objectives Understood;
- Manage Expectations;
- Service Planning;
- Measuring service level achievements;
- Customer Satisfaction

Availability Management

- Proactive, high level strategic;
- Cross-function;
- Awareness and communication;
- Process and Architecture Governance;
- Improvement initiatives;
- Value of Availability

Event Management

- Define and govern Monitoring Strategy
- Define Monitoring Architecture;
- Implement monitoring tools;
- Correlate component and service events
- Automate responses
- Continuously tune thresholds

Configuration Management

- Component Location;
- Connectivity and linkages;
- Contacts;
- Business Impact;
- Index to recovery procedures;
- Horizontally and vertically integrated DB

Problem Management

- Quick analysis (Cause of incident) and timely correction;
- Track corrections and escalate exceptions;
- Reduce recurring incidents;
- Knowledge DB;
- Problem Prevention - True root cause (Cause of defect);
- Pursuit of secondary contributors;
- Failure Pattern and Trend Analysis

Testing

- System volume and stress;
- Production-like platform and data;
- Test scripts and Transaction Driver;
- Failure injection and recovery testing

Recovery Development

- Understand potential failures;
- Develop Problem Determination (PD) tools;
- Develop Recovery procedures;
- Automate recovery;
- Test and practice recovery (fire drills)

Component Failure Impact Analysis

- Methodology to provide IT with insight as to how deficiencies in the infrastructure, processes, procedures, and service delivery skills impact the business operation. Helps identify:
 - Need for additional resilience to minimize the impact of a component failure on business services
 - Need for enhancements to procedures, processes, and skills
- Goals of CFIA
 - Identify critical availability and performance risks for business services
 - Identify measures to reduce probability of failure
 - Identify measures to reduce time to detect and repair failures

An Architecture for Achieving Highly Available Business Services

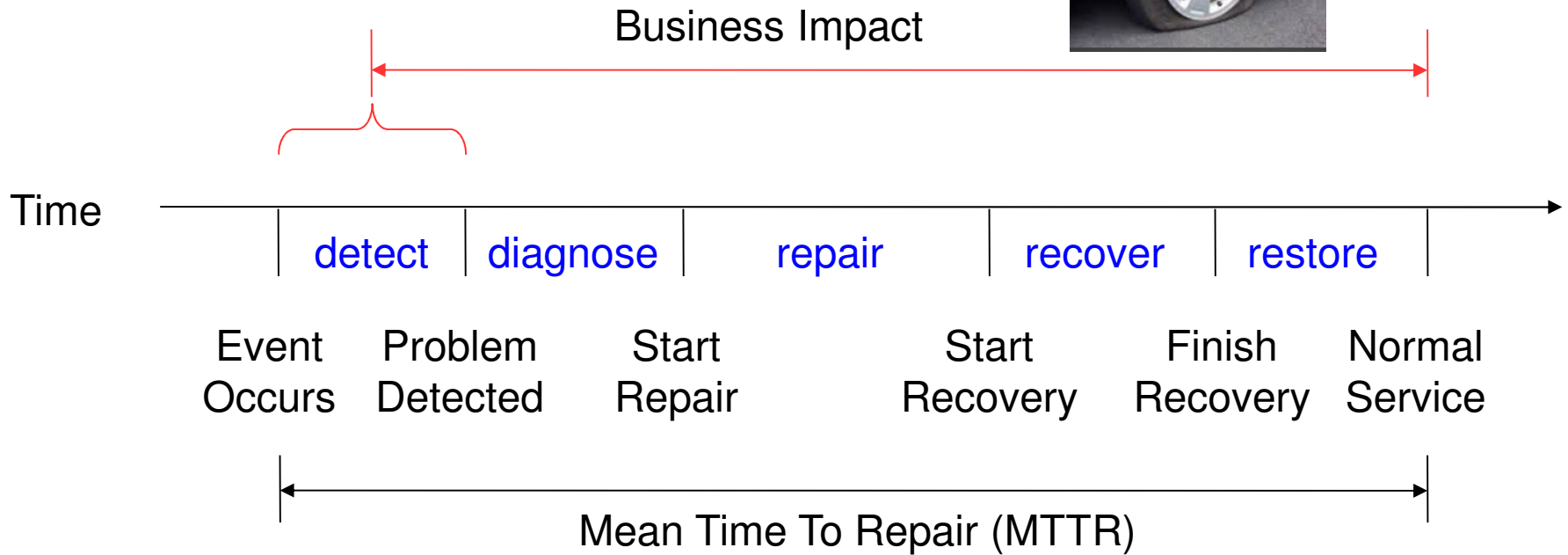
- Reliable - minimize chance of failure
- Resilient - minimize chance of service outage
- **Recoverable** - **minimize duration of service outage**
- Resourceful - minimize the damage

If failure cannot be masked, business services will be disrupted. Now we must seek to minimize the duration of the disruption ...

Minimize service outage by reducing MTTR ?



Let's go to work.



You may quibble with my endpoints

- This is a typical approach to resolving failures
- But this is too slow for HA !

Fast recovery minimizes duration of disruption to business services

Ideally, a fault tolerant architecture and infrastructure allows service to continue uninterrupted despite component failure.



Not all failures will be masked. Rapidly restoring service minimizes the business impact.



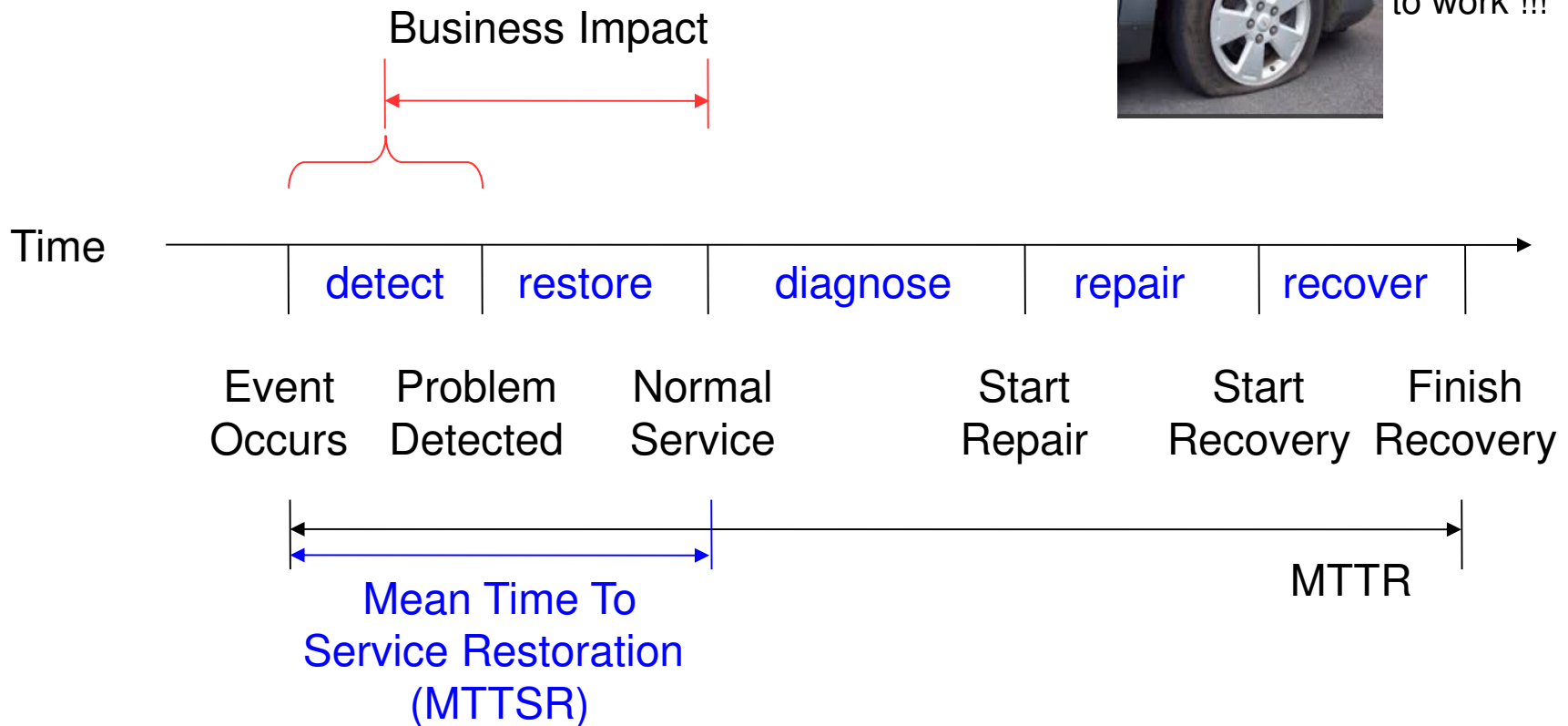
- The key objective for fast recovery is to restore normal operation of the business service as quickly as possible
- Diagnosis and repair are secondary issues to be addressed later
- People often need to change their mindset

Depending on your SLA's, fast recovery might achieve "no disruption"

Use fast recovery to minimize service outage



I must get to work !!!



The number one priority is to restore normal business operation ASAP.
Everything else is secondary.

But how can I restore service if there is failed component ...

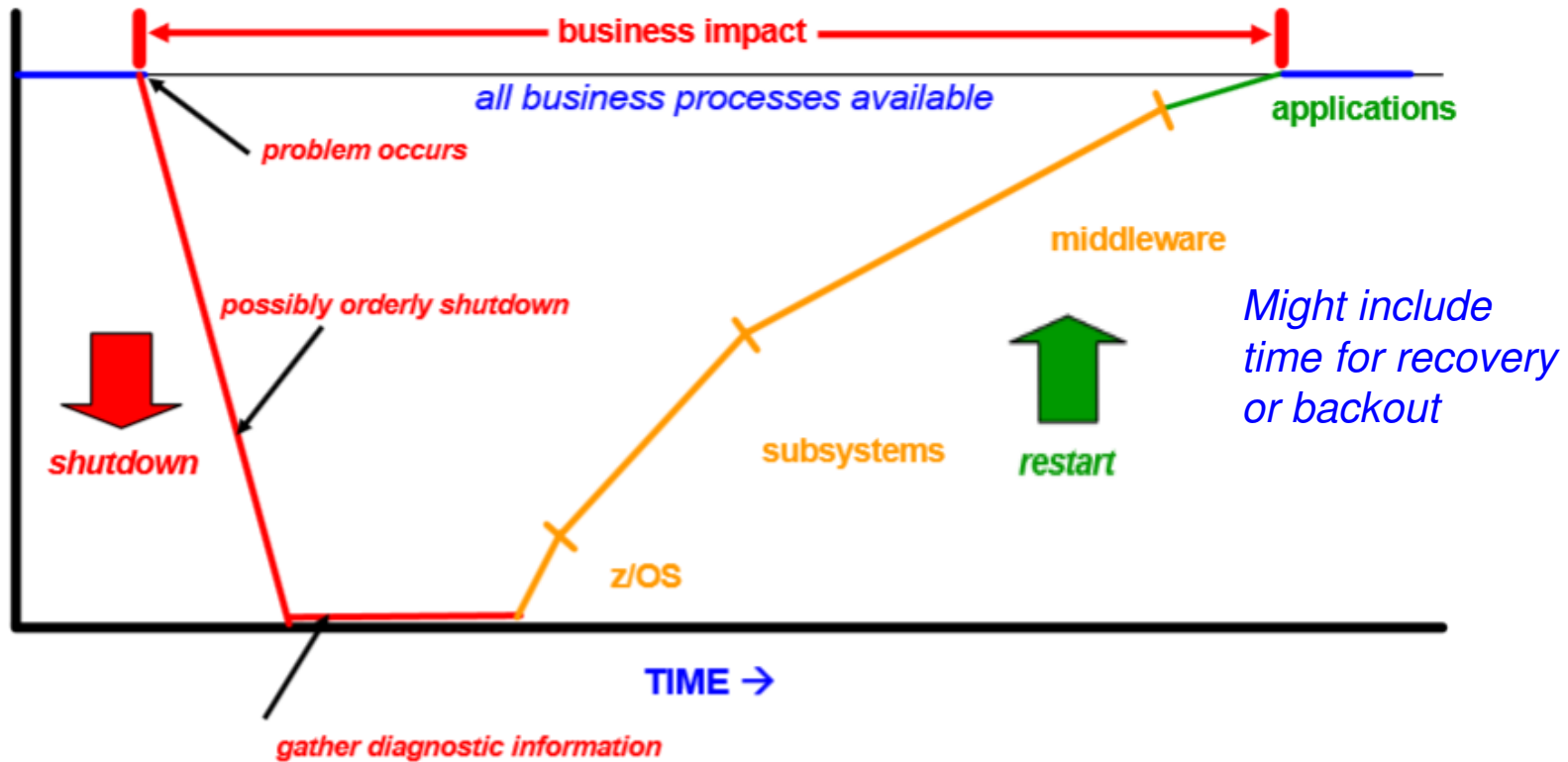
Restoring Service

- Get the service running again somewhere. Perhaps:
 - Hot standby
 - Rapid restart
- Service restoration strategy likely impacted by:
 - Specific capabilities/restrictions imposed by the application
 - Inter-dependencies of the components that need to be recovered
 - Robustness of the architecture and its implementation
 - Need for transaction recovery or back out, etc
- If possible, build one restoration method appropriate to the business service. Use it exclusively.
 - Simplicity and familiarity foster reliability of the restoration process
 - Continuous improvement to shorten; move to sysplex enablement
 - Choice of methods implies diagnosis; increases cost and complexity

Recovery and restoring service

- After a failure, some sort of recovery processing might be needed in order for surviving instances to continue
- Peer based recovery
 - The surviving peer instances perform the required recovery processing
 - If so, rapid restoration of the failed instance is not critical for maintaining the business service
 - Though you might consider such to restore normal operating conditions and reduce risk of impact from a subsequent failure
- Restart-based recovery (for example, DB2)
 - Failed instance must be restarted to perform its own recovery processing (run logs, resolve in-flight transactions, drop locks)
 - Here, automation is critical for rapid recovery

Repair time for system restart



Restart of a system is a long repair time, so an unlikely choice. The point: model the timeline and look for opportunities to shorten it.

Recoverable: Minimize duration of outage

- Fast recovery requires
 - Comprehensive monitoring
 - Automated detection
 - Automated recovery
- Knowledge
 - What can fail?
 - What will it impact?
- Testing
 - Failure injection
 - Validation of recovery
- Robust technology
 - Some place to go
 - Application enablement

Service Level Management

- Business Requirements;
- Common IT Service Level Objectives Understood;
- Manage Expectations;
- Service Planning;
- Measuring service level achievements;
- Customer Satisfaction

Availability Management

- Proactive, high level strategic;
- Cross-function;
- Awareness and communication;
- Process and Architecture Governance;
- Improvement initiatives;
- Value of Availability

Event Management

- Define and govern Monitoring Strategy
- Define Monitoring Architecture;
- Implement monitoring tools;
- Correlate component and service events
- Automate responses
- Continuously tune thresholds

Configuration Management

- Component Location;
- Connectivity and linkages;
- Contacts;
- Business Impact;
- Index to recovery procedures;
- Horizontally and vertically integrated DB

Problem Management

- Quick analysis (Cause of incident) and timely correction;
- Track corrections and escalate exceptions;
- Reduce recurring incidents;
- Knowledge DB;
- Problem Prevention - True root cause (Cause of defect);
- Pursuit of secondary contributors;
- Failure Pattern and Trend Analysis

Testing


- System volume and stress;
- Production-like platform and data;
- Test scripts and Transaction Driver;
- Failure injection and recovery testing

Recovery Development

- Understand potential failures;
- Develop Problem Determination (PD) tools;
- Develop Recovery procedures;
- Automate recovery;
- Test and practice recovery (fire drills)


Installations can significantly improve service availability by focusing on reducing recovery time

- Service disruptions result from a combination of three factors:
 - n = number of incidents (component reliability affects frequency of failures – MTBF)
 - d = duration of incidents (recoverability of service affects length of down time – MTTR)
 - s = scope of impact (restrictability of number of users, transactions, services impacted).



Recoverability has the most influence on service availability

$$\text{Service Disruption Index} = \sum_{i=1}^n d_i * s_i$$



“If you plan for no failures, you will fail!”

- Getting to 100% service availability requires only that any one factor be zero.
 - Number of incidents will never be zero
 - In large, highly shared environments, scope of impact can be difficult to control
 - By masking business services from component failures through fast recovery, we have a good chance of moving the duration of service down-time to near zero.
- Minimizing MTTR provides the greatest benefit to service availability and is the one factor where IT has the most influence.

An Architecture for Achieving Highly Available Business Services

- Reliable - minimize chance of failure
- Resilient - minimize chance of service outage
- Recoverable - minimize duration of service outage
- **Resourceful - minimize the damage**

If business services have been disrupted and we cannot recover in a timely fashion, we are in crisis. We must now seek to minimize the damage...

re·source·ful – adjective

- 1. able to deal well with new or difficult situations and to find solutions to problems
 - 2. able to meet situations; capable of devising ways and means
 - 3. having the ability to find quick and clever ways to overcome difficulties
-
- Talented, skilled, knowledgeable, clever, creative people are a tremendous asset to any organization. In the context of delivering highly available services, we want the bulk of their efforts directed towards making the IT services reliable, resilient, and recoverable. But when we do suffer an outage, we need them to resolve the problem as quickly as they can.

We do not want to be here

- Our attempts to make the system reliable, resilient, and recoverable have failed
 - Something needs to be fixed so this won't happen again
- There are really two issues:
 - The failure of the system component
 - The failure to mask that failure from the user of the business service
- Good service management processes will ensure that we:
 - Get to root cause of both failures with due consideration of all contributing factors
 - Technology, Processes, Procedures, Skills
 - Drive corrective actions to avoid/mitigate future service disruptions
 - Improve reliability, resilience, recoverability of the system
 - Improve ability to resolve issues through manual intervention in a more timely manner

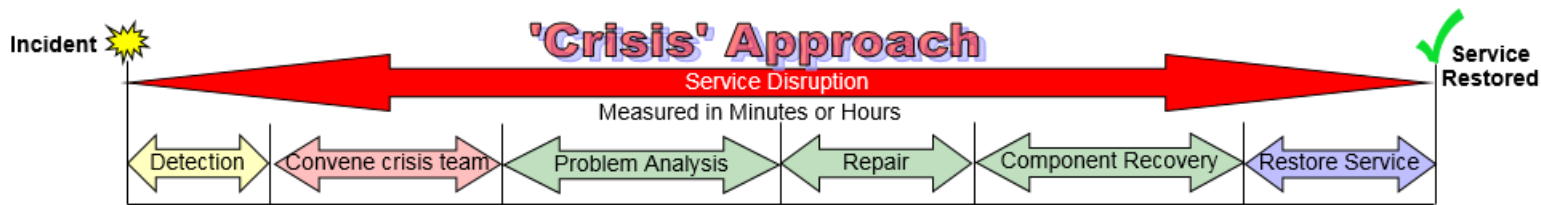
But for now, we have a serious situation to address

In a crisis we must be resourceful

Ideally, a fault tolerant architecture and infrastructure allows service to continue uninterrupted despite component failure.

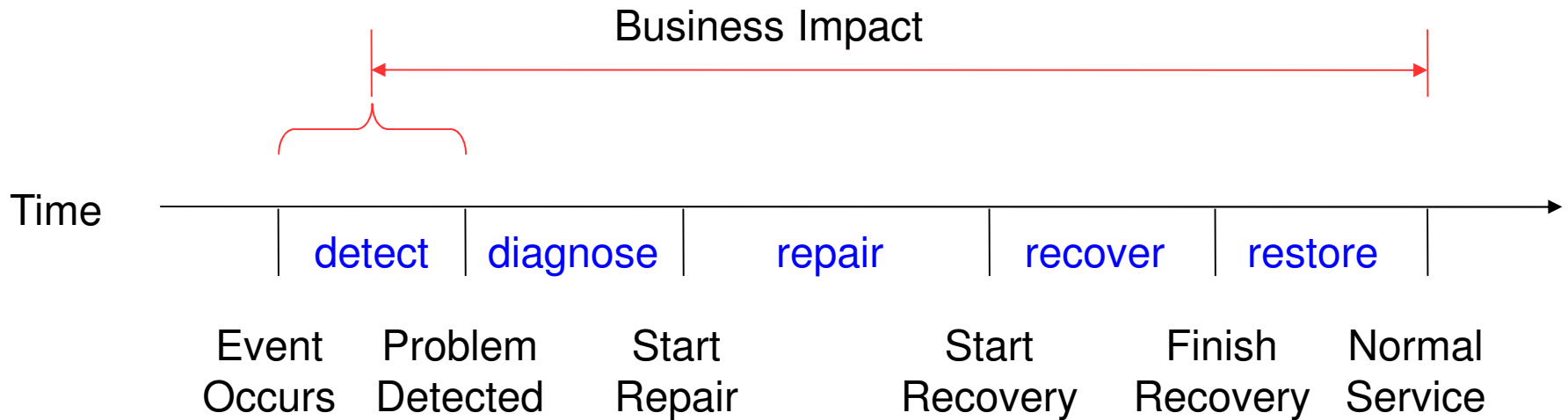


Not all failures will be masked. Rapidly restoring service minimizes the business impact.



When service restoration requires human intervention, outages tend to have unpredictable (long) duration. Risks significant business impact.

Be prepared to minimize duration of business impact through manual intervention



Ensure that staff has necessary skills, training, and resources:

- Experience dealing with failures
- System knowledge
- Documented procedures
- Rehearsed procedures
- Failure testing

Consider ways for the system to help the staff resolve the issue quickly:

- Comprehensive monitoring
- Automated alerts
- Automated recovery procedures
- Diagnostic tools

Resourceful: Minimize the damage

- Incident management
 - Manage stakeholders
 - Assist staff
- Knowledge
 - Normal? Changed?
 - Broke? Impact?
- Automation
 - Alerts
 - Recovery
- Be ready
 - Failure testing
 - Practice recovery

Incident Management

- Fast Service Restoration;
- Prevent crises;
- Operational skills;
- PD tools and procedures

Configuration Management

- Component Location;
- Connectivity and linkages;
- Contacts;
- Business Impact;
- Index to recovery procedures;
- Horizontally and vertically integrated DB

Event Management

- Define and govern Monitoring Strategy
- Define Monitoring Architecture;
- Implement monitoring tools;
- Correlate component and service events
- Automate responses
- Continuously tune thresholds

Testing

- System volume and stress;
- Production-like platform and data;
- Test scripts and Transaction Driver;
- Failure injection and recovery testing

Service Level Management

- Business Requirements;
- Common IT Service Level Objectives Understood;
- Manage Expectations;
- Service Planning;
- Measuring service level achievements;
- Customer Satisfaction

Change Management

- Enable growing quantity of changes;
- Measure Effectiveness by function and educate;
- Risk assessment and mitigation;
- Exception handling for emergencies and business needs;
- Accountability lies with developer;
- Readiness Checklists and implementation scripts;
- Focus on Quality of Change itself

Problem Management

- Quick analysis (Cause of incident) and timely correction;
- Track corrections and escalate exceptions;
- Reduce recurring incidents;
- Knowledge DB;
- Problem Prevention - True root cause (Cause of defect);
- Pursuit of secondary contributors;
- Failure Pattern and Trend Analysis

Recovery Development

- Understand potential failures;
- Develop Problem Determination (PD) tools;
- Develop Recovery procedures;
- Automate recovery;
- Test and practice recovery (fire drills)

Availability Management

- Proactive, high level strategic;
- Cross-function;
- Awareness and communication;
- Process and Architecture Governance;
- Improvement initiatives;
- Value of Availability

Root cause analysis for HA

- **Must be comprehensive**
 - What factors allowed this problem to cause this impact?
 - Consider all aspects of architecture, technology, processes, and skills

- **Must result in corrective action**
 - What changes will we make to prevent this problem from occurring in production in the future?
 - What changes will we make to minimize the impact of this problem if it does reoccur in production?
 - Detect the problem sooner?
 - Minimize the scope of the impact?
 - Minimize the duration of the impact?
 - In other words, how can we make the system more reliable, resilient, and recoverable

Inhibitors to achieving HA

- Unable to make business justification
 - HA requires investment in both technology and people
 - The spending averse are reluctant to incur hard costs today to avoid the costs associated with future outages that might never occur
 - Need better linkage between IT spending and business goals
- Culture
 - Assumption that technology in and of itself provides HA
 - Failure to be proactive
 - Inadequate processes and procedures
- Applications
 - Not sysplex enabled
 - Affinities, lack of coexistence, “rolling IPL”, ...

Inhibitors to achieving HA

- Redundancy issues
 - Inadequate redundancy
 - Ineffective redundancy

- Reliance on manual intervention
 - Inadequate automation
 - Failure to exploit autonomic features

Cost of an outage?

- Expected Value of the loss from an outage is a function of the cost of the outage times the risk that the outage occurs
 - $EV(\text{outage}) = \text{Cost}(\text{outage}) * \text{Risk}(\text{outage})$

- Quantifying Cost
 - Penalty for missed SLA's
 - Loss of productivity
 - Loss of revenue
 - Customer dissatisfaction, loss of confidence
 - Brand reputation

Can be difficult to quantify.
Time dependencies?

- Quantifying Risk
 - MTBF, MTTR, reliability

Can be difficult to quantify.
Lack data or too complex.

- Use past outages as a guide?

An Architecture for Achieving Highly Available Business Services

- Reliable - Minimize number of incidents
- Resilient - Mask failures that do occur
- Recoverable - Minimize disruption of unmasked failures
- Resourceful - Be prepared when all else fails

Layered protection built on:

- The premise that failures will occur
- A robust, fault tolerant architecture
- Well implemented technology
- Excellent processes and procedures
- Skilled staff
- Well understood objectives

For more information

- **Forbes: Reputational Impact of IT Risk**
 - <http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=>
- **Aligning IT with Strategic Business Goals**
 - http://www.ibm.com/midmarket/it/it/att/pdf/it_IT_business_continuity_BCRS
- **Redbooks**
 - 2004 Achieving the Highest Levels of Parallel Sysplex Availability (SG24-6061)
 - 2004 Parallel Sysplex Application Considerations (SG24-6523)
 - 2010 System z Mean Time to Recovery Best Practices (SG24-7816)
 - 2011 System z Parallel Sysplex Best Practices

Please complete your session evaluation

Parallel Sysplex: Achieving the promise of high availability
Session 17431

Complete your session evaluations online at www.SHARE.org/Orlando-Eval